UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Solução Mobile - Banking Pagamentos

Rodrigo Manuel Torres Rodrigues

Mestrado em Engenharia Informática
Especialização em Engenharia de Software

Trabalho de Projeto orientado por:
Prof. Doutor Marco Giunti
Ricardo Filipe Baptista Nogueira

2018

Agradecimentos

Agradeço aos meus pais por tudo o que fizeram por mim, por todo o apoio e amor que me deram. Sem eles nada seria possível.

À minha irmã por todo o apoio que me deu e por ser alguém que me inspira pela sua força.

À minha namorada por estar sempre ao meu lado, por me ter ajudado a corrigir os erros deste relatório e a pô-lo com bom aspeto, por tudo o que me dá sem esperar nada em troca.

Aos meus avós e resto da família que estão sempre disponíveis para o que for preciso.

A todos o meus amigos que me apoiaram ao longo desta difícil fase.

Aos meus orientadores pela ajuda que me deram.

Aos meus pais.

Resumo

Na situação em que o mercado se encontra, é essencial para uma empresa ter uma solução móvel para o seu serviço. Um dos ramos empresariais que tem apostado fortemente nestas soluções são os bancos. Estes tentam facilitar ao máximo o acesso dos seus clientes a serviços como pagamentos, transferências, saldos, movimentos, entre outros, sendo esta uma maneira de manter os mesmos satisfeitos.

Existem consultoras com projetos que visam a conceção de soluções para estas empresas. Uma destas consultoras é a *Accenture*, correspondendo à empresa onde será efetuado todo o trabalho descrito neste relatório.

No decorrer deste estágio foi concebida uma solução mobile de *homebanking* direcionada para a funcionalidade de pagamentos.

Numa fase inicial foi feito um enquadramento do tema a partir de uma investigação, onde também foram analisados os requisitos funcionais e não funcionais, assim como os casos de uso, a metodologia e os recursos a serem utilizados.

De seguida foi feito o desenho arquitetural do sistema, da base de dados, foi definida a navegação na aplicação e foi escolhido o padrão arquitetural a utilizar.

A terceira fase consistiu na implementação da solução, tendo sido implementadas todas as funcionalidades da aplicação nativa para o sistema operativo *Android*, uma base de dados *SQL Server* que implementa o desenho falado anteriormente, e um servidor web implementado a partir da *framework Spring* que intermedeia os dois últimos fornecendo serviços à aplicação *Android* e efetuando operações na base de dados. No fim desta implementação, o código que se encontrava na linguagem Java foi migrado para a linguagem *Kotlin*, que recentemente foi oficializada como uma linguagem para o desenvolvimento *Android*.

Por fim, foram feitos vários testes à aplicação, incluindo testes de usabilidade, de compatibilidade e foram feitos testes de maneira a poder comparar uma funcionalidade inovadora com uma já existente no mercado.

Palavras-chave: *Banking*; Aplicação Nativa; *Android*; Pagamentos

Abstract

In the actual market situation, is essential for a company to have a mobile solution for their services. One of the enterprise sectors that has been betting strongly on that solutions are the banking companies. To make their client satisfied, they try to make the access to their services, like payments, transfers, balances, account moves and so on, as easy as possible.

There are consultant companies with projects that aim the conception of solutions to companies like banks. One of that consultants is Accenture, the company where all the work described in this report will be made.

During this internship an homebanking mobile solution directed to payments functionality was concept.

On an initial stage, was made a subject framework based on an investigation, the functional and non-functional requirements were analyzed as well as use cases, work methodology and the resources to be used.

Then the system architecture, database and navigation design was made and the architectural pattern to apply was chosen.

The third stage consisted in the solution implementation where all the functionalities of the Android native app were implemented as well as a SQL Server database and a Spring web server that provides services to the mobile app and makes database operations. In the end of that implementation the Java code of mobile app was migrated to Kotlin code, a language that was recently formalized as an Android developing language.

Finally, some mobile app tests were made, including usability and compatibility tests. Also, were made some tests to compare an innovative functionality with another that already existing on the market.

Keywords: Banking; Native App; Android; Payments

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho encontra-se enquadrado na unidade curricular Projeto de Engenharia Informática, estando esta incluída no segundo ano do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa. Para tal, foi realizado um estágio com a duração de 9 meses numa instituição de acolhimento com o intuito de ganhar conhecimentos a nível profissional.

Neste capítulo será descrita a instituição de acolhimento assim como a motivação e os objetivos deste trabalho. Por fim, será apresentada a estrutura do documento.

1.1 *Accenture*

O estágio foi realizado na *Accenture*, uma empresa internacional que fornece serviços nas áreas de estratégia, consultoria, digital, tecnologia e operações. Possui clientes em mais de 120 países, trabalhando em mais de 40 industrias e contando com cerca de 425 000 empregados, sendo que 2 000 corresponde ao número de funcionários em Portugal.

1.2 Motivação

As aplicações móveis têm vindo a evoluir cada vez mais levando às empresas a procurarem por este tipo de soluções tecnológicas. Um exemplo desta evolução pode ser verificada na industria bancária, onde se tem desenvolvido aplicações que pretendem facilitar a realização de operações bancárias, que antes só poderiam ser efetuadas numa caixa de multibanco. Estas aplicações permitem a realização de operações como: a consulta do saldo e/ou movimentos da conta; a realização de pagamentos e/ou transferências; o agendamento de pagamentos e/ou transferências.

1.3 Objetivos

O objetivo inicial deste estágio consistia na realização de uma aplicação móvel nativa para o sistema operativo *Android*, com vista a atualizar a aplicação corrente de um banco cliente. O projeto de desenvolvimento desta aplicação seria realizado em conjunto com uma equipa de trabalho. Contudo, devido a problemas internos o projeto não entrou em execução, sendo necessário a definição de um novo projeto de modo a finalizar o estágio.

A solução encontrada consistiu em realizar um projeto idêntico de forma individual. Esta resume-se ao desenvolvimento de uma aplicação móvel bancária focada na funcionalidade de pagamentos. Neste desenvolvimento serão compreendidas todas as fases de conceptualização do software, sendo elas as seguintes:

- Levantamento de requisitos;
- Análise funcional, técnica e arquitetural do sistema;
- Desenho funcional;
- Desenvolvimento de protótipos;
- Implementação técnica da solução;
- Configuração da solução integrada;
- Realização de testes integrados.

1.4 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Conceitos e trabalho relacionado
- Capítulo 3 – Análise de requisitos
- Capítulo 4 - Desenho da solução
- Capítulo 5 - Implementação
- Capítulo 6 - Testes
- Capítulo 7 - Conclusão

Capítulo 2

Conceitos e Casos de Estudo

Este capítulo será divido em duas partes. Na primeira serão introduzidos alguns conceitos importantes no decorrer deste trabalho e, na segunda parte, serão apresentadas algumas aplicações semelhantes.

2.1 Conceitos

Nesta secção serão apresentados alguns conceitos relacionados com a indústria bancária e com o software utilizado.

2.1.1 Pagamentos

A operação pagamento é algo de conhecimento comum, constitui a possibilidade de um cidadão com uma conta bancária efetuar a transferência de um certo montante para uma determinada entidade. Estas transferências contêm uma referência que serve para identificar a fatura a que o pagamento se refere e uma entidade que identifica o destinatário do pagamento. Existem vários tipos de pagamento [25] consoante a entidade destinatária. Estes serão descritos separadamente de seguida.

Pagamento de serviços

Possibilita o pagamento de faturas a entidades como empresas, instituições e organizações. Para realizar este pagamento basta introduzir a entidade, a referência e o montante presentes na fatura.

Pagamento ao estado

Possibilita o pagamento de impostos, taxas, custas jurídicas, juros de mora, penhoras, entre outros. Para efetuar este pagamento basta introduzir a referência e o montante presentes na fatura.

Pagamento à segurança social

Possibilita a realização de um pagamento à segurança social. Para efetuar este pagamento basta introduzir o número de beneficiário de segurança social, a remuneração e o período de pagamento. Após a inserção dos dados anteriores, o montante final do pagamento é calculado automaticamente de acordo com o algoritmo da segurança social. Este algoritmo será explicado mais à frente na Secção 5.1.2.

2.1.2 *Android*

O *Android* é o sistema operativo para plataformas móveis mais popular do mundo. Centenas de milhões de dispositivos, em cerca de 190 países diferentes, utilizam esta plataforma. De acordo com as estatísticas [28], em 2017, na Europa, apenas 2% dos *smartphones* possuem o sistema operativo *Windows Phone*, 30% o sistema operativo *iOS* e 68% o sistema operativo *Android*.

O *Android* é um sistema operativo gratuito baseado no sistema operativo *Linux*. O seu produtor foi a *Open Handset Alliance*, que consiste num conjunto de empresas tecnológicas lideradas pela Google e, foi concebido para sistemas móveis, tais como *smartphones*, *tablets* ou *smartwatches*.

Android Software Development Kit

Android SDK é um conjunto de ferramentas de desenvolvimento e *debugging*, emuladores, ferramentas de automatização e compilação, bibliotecas, entre outras. Estas ferramentas são utilizadas para simplificar todo o processo desenvolvimento de aplicações nativas para o sistema operativo *Android*. O SDK será utilizado sobre o ambiente de desenvolvimento (*IDE*) oficial da Google, o *Android Studio* [14], que é baseado na *IDE IntelliJ*.

Nível de API

O nível de API corresponde a um valor inteiro referente a cada versão da plataforma *Android*. Uma aplicação destinada a um certo nível de API pode correr normalmente em plataformas com níveis superiores, mas pode vir a ter problemas em plataformas com níveis inferiores.

Na Tabela 2.1 são apresentadas as várias versões existentes do sistema operativo *Android*, através do respetivo número e nome. Para cada versão existente é também apresentado o respetivo nível de API e a percentagem de dispositivos com essa mesma versão. Os dados da Tabela foram coletados durante um período de 7 dias entre os dias 1 e 8 de Janeiro de 2018. Todas as versões instaladas em menos de 0,1% dos dispositivos não são exibidas.

Tabela 2.1: Versões da plataforma [16]

Versão	Nome	API	Distribuição
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%

A partir da Tabela 2.1 observa-se que 10.3% dos dispositivos *Android* correm sobre a API 19,

verificando-se que uma aplicação destinada a esta API tem a capacidade de correr em 95% ($10.3\% + 4.8\% + 17.6\% + 25.5\% + 22.9\% + 8.2\% + 4.9\% + 0.8\% = 95\%$) dos dispositivos *Android* sem o risco de ocorrer uma falha relacionada com a API do dispositivo.

2.1.3 Tipo de Desenvolvimento

Nesta secção serão introduzidas os dois tipos de desenvolvimento de uma aplicação móvel, desenvolvimento nativo e desenvolvimento *Cross-Platform*, bem como algumas comparações entre estes dois tipos.

Desenvolvimento Nativo

As aplicações nativas são desenvolvidas numa linguagem específica para cada plataforma e usadas exclusivamente no sistema operativo escolhido. No caso do sistema operativo *Android* é utilizada a linguagem *Java* ou *Kotlin* e, no caso do *iOS*, é utilizada a linguagem *Swift*.
Alguns aspetos positivos deste tipo de desenvolvimento são os seguintes:

Interface: A interface com o utilizador pode determinar o sucesso de uma aplicação móvel. Numa aplicação nativa a interface é concebida num padrão que é familiar ao utilizador, pois cada sistema operativo tem características individuais, algo que é facilmente percetível pelo seu utilizador. As aplicações nativas seguem essas características e por isso tornam mais fácil a sua utilização;

Performance: O desenvolvimento nativo, quando bem executado, otimiza as aplicações mais densas em termos de custos computacionais. Isto acontece porque a linguagem utilizada interage diretamente com o sistema operativo evitando assim que a aplicação se torne lenta e penosa;

Melhor Classificação: As aplicações nativas normalmente encontram-se melhor classificadas nas *app stores*. Isto acontece porque tanto a performance como a interface fazem com que os utilizadores forneçam melhores cotações às mesmas;

Conexão à Internet: Dependendo da funcionalidade, uma aplicação nativa consegue trabalhar *offline*.

Enquanto que algumas desvantagens são as seguintes:

Tempo de Desenvolvimento: O processo de desenvolvimento nativo é lento, pois é necessário desenvolver mais do que uma aplicação caso seja pretendido que esta seja executável em diferentes plataformas;

Necessários Programadores Especializados: Para o desenvolvimento de uma aplicação nativa a especialização dos programadores é um fator importante, isto porque vão haver equipas de desenvolvimento diferentes para cada plataforma. Por exemplo, será necessário uma equipa para desenvolver a aplicação para a plataforma *Android*, uma equipa para desenvolver a aplicação para *iOS*, entre outras, se necessário. Caso isto não aconteça o tempo de desenvolvimento vai crescer exponencialmente;

Preço: Os custos no aumento do número de programadores necessários para cada plataforma juntamente com o tempo despendido no desenvolvimento das mesmas, irão fazer com que o custo final de desenvolvimento da aplicação seja muito superior.

Desenvolvimento *Cross-platform*

As aplicações *cross-platform* são desenvolvidas numa linguagem capaz de se adaptar a vários sistemas operativos, através do uso de *frameworks*. Esta abordagem permite que uma única versão da aplicação seja executada em vários sistemas operativos diferentes. A maioria das aplicações *cross-platform* são desenvolvidas em linguagens como *HTML5*, *CSS*, *JavaScript*, *C#*, entre outras. Estas são transformadas em aplicações nativas utilizando *frameworks* como o *Cordova* ou o *Xamarin*.
Algumas vantagens deste tipo de desenvolvimento são os seguintes:

Tempo de Desenvolvimento: Como apenas é concebida uma aplicação para os vários sistemas operativos, o tempo de desenvolvimento desta aplicação será reduzido;

Preço: Como não é necessária uma equipa tão especializada e o tempo de desenvolvimento é reduzido, o custo de desenvolvimento torna-se mais acessível;

Fácil Atualização: Novamente, como apenas existe uma aplicação, basta atualizar a mesma.

Enquanto que alguns pontos negativos do desenvolvimento *Cross-platform* são os seguintes:

Performance: A performance da aplicação sai prejudicada devido às restrições no que toca ao poder computacional dos dispositivos móveis. A renderização de páginas *HTML5/CSS* pesadas pode fazer com que a aplicação se torne lenta em dispositivos com CPU's ou GPU's de gamas mais baixas;

Limitações: Uma aplicação *cross-platform* não está preparada para funcionar com tudo o que um sistema operativo oferece. Desta forma, não é possível tomar todo o proveito de algumas funcionalidades tais como o GPS, a câmara ou o calendário, nestas aplicações.

Comparação

As aplicações desenvolvidas em *cross-platform* têm dois grandes fatores a seu favor: o tempo de desenvolvimento e o custo de produção. Apesar disso, no que diz respeito à experiência de utilização esta fica a perder, em relação ao outro tipo de desenvolvimento. A experiência de utilização é um fator crítico para o sucesso de uma aplicação, uma fraca experiência de utilização pode ser o suficiente para que o utilizador desinstale a aplicação. Um estudo realizado pela *Compuware* [27] concluiu que apenas 7% dos utilizadores daria mais do que quatro oportunidades a uma aplicação com falhas. Assim, é possível concluir que a experiência de utilização de uma aplicação é diretamente proporcional ao custo da mesma.

Desta forma, apesar de uma aplicação nativa trazer custos e tempos de desenvolvimento superiores, a performance e a experiência de utilização da mesma serão melhores face ao outro tipo de desenvolvimento. Assim, uma aplicação nativa será a melhor escolha quando os objetivos são satisfazer o utilizador.

2.1.4 Padrão Arquitetural

Para que uma aplicação cumpra alguns requisitos de qualidade, esta deve ser bem estruturada seguindo um padrão arquitetural. Existem vários padrões arquiteturais, sendo que os padrões mais utilizados serão explicados de seguida. O padrão arquitetural utilizado na implementação da aplicação desenvolvida será descrito na Secção 4.2.

Model-View-Controller

O padrão *Model-View-Controller* separa a aplicação em três camadas. Na primeira camada, camada *Model*, é feita a persistência da aplicação e toda a manipulação de dados. Numa segunda camada, denominada *View*, é a camada de visualização, onde se encontram os *layouts* que serão vistos pelo utilizador. Por último, na camada *Controller*, são recebidos todos os pedidos do utilizador e manipuladas as outras duas camadas. Na Figura 2.1 são representadas as comunicações entre as três camadas mencionadas anteriormente.

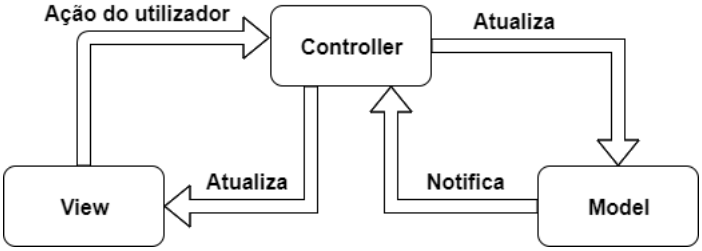


Figura 2.1: Diagrama *Model-View-Controller*

Model-View-ViewModel

Model-View-ViewModel é um padrão que separa também a aplicação em três camadas. A primeira é o *Model*, esta camada é responsável pela manipulação dos dados e pela lógica de negócio da aplicação. A camada *View* é basicamente a interface com o utilizador. A última camada, denominada de *ViewModel*, é responsável por realizar um modelo da camda *View*, isto é, readquire os dados necessários da camada *Model* e utiliza-os na camada *View* para que se torne visível ao utilizador. Na Figura 2.2 são representadas as comunicações entre as três camadas mencionadas anteriormente.

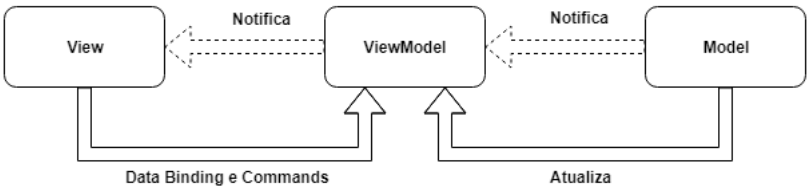


Figura 2.2: Diagrama *Model-View-ViewModel*

Model-View-Presenter

O padrão *Model-View-Presenter* separa, mais uma vez, a aplicação em três camadas. A primeira camada, camada *Model*, é responsável por manipular a lógica de negócio e comunicação com a rede e com a base de dados. A camada *View* é responsável por toda a interface com o utilizador e também notifica a última camada, denominada de *Presenter*, com as ações do utilizador. A camada *Presenter* é responsável por restabelecer os dados da camada *Model*, aplicar a lógica de interface com o utilizador e gerir o estado da camada *View*, decidindo o que é disponibilizado ao utilizador. Na Figura 2.3 são representadas as comunicações entre as três camadas mencionadas anteriormente.

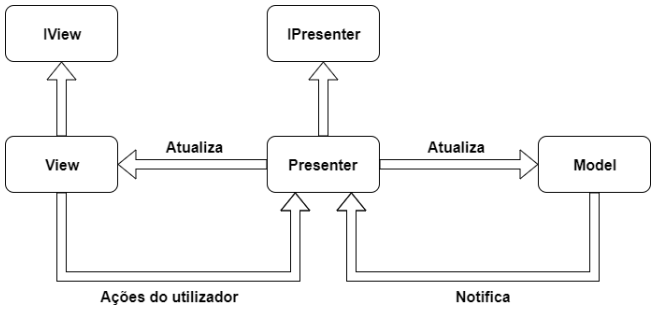


Figura 2.3: Diagrama *Model-View-Presenter*

2.1.5 *SQL Server*

SQL Server é um sistema de gestão de base de dados concebida pela Microsoft. Este sistema suporta uma grande variedade de processos transacionais, aplicações de análise de dados e *business intelligence*. O sistema trabalha sobre a linguagem *T-SQL* (*Transact-Structured Query Language*), sendo uma implementação feita pela Microsoft que adiciona um conjunto de propriedades à linguagem *SQL* padrão.

2.1.6 2FA

Para que a autenticação de uma conta bancária numa aplicação de *homebanking* seja segura, esta tem de satisfazer os requisitos de 2FA (*two-factor autenticaton*), ou seja, tem de possuir dois fatores de autenticação. Existem várias formas de autenticação, sendo que estas podem ser distribuídas por 3 categorias [30], conhecidas como fatores de autenticação. Os três fatores são os seguintes:

1. **Fator de conhecimento:** algo que o utilizador sabe, tal como, uma *password* ou um *PIN*;
2. **Fator de posse:** algo que o utilizador possui, tal como, um cartão magnético, um *token* de segurança ou uma matriz de valores;
3. **Fator de inerência, ou biometria:** algo que o utilizador é, ou seja, características físicas ou comportamentais exclusivas, como por exemplo, impressão digital, reconhecimento facial ou de voz.

2.2 Casos de Estudo

No desenvolvimento deste projeto foram analisadas outras aplicações da mesma área como base para a aplicação desenvolvida. Todas as aplicações analisadas correspondem a aplicações móveis de bancos Portugueses, sendo que a informação acerca destes está disponível na *Google Play Store* e nos sites dos respetivos bancos. Destas aplicações serão analisadas essencialmente as interfaces, a funcionalidade pagamentos e o fluxo da mesma.

Caixa Geral de Depósitos: Caixadirecta

Esta aplicação possui 4 sub-funcionalidades de pagamento [2]:

- Pagamento de serviços: o utilizador começa por selecionar uma conta e, de seguida preenche os campos referentes à fatura, sendo estes “Entidade”, “Referência” e “Montante”. Neste tipo de

pagamento também possível fazer o agendamento de um pagamento consoante uma certa periodicidade[1] ou para um determinado dia;

- Pagamento de Telecomunicações/Internet/Transportes: muito semelhante ao pagamento anterior mas ao invés do utilizador preencher o campo "Entidade", seleciona um operador de telecomunicações, Internet ou transportes;
- Pagamento de impostos/segurança social: para o pagamento de impostos o utilizador preenche uma referência com 15 caracteres e o montante. Neste também é possível selecionar a conta e a data do pagamento. Num pagamento à segurança social o utilizador deve preencher os campos "Número de beneficiário", a remuneração mensal ou o número de dias ou horas de trabalhado. De seguida, selecionando o ano e o mês, o montante final é calculado automaticamente. Este pagamento pode também ser agendado para um determinado dia;
- Pagamento de cartões: esta funcionalidade é utilizada para liquidar um determinado cartão de crédito que esteja associado à conta. Nesta funcionalidade seleciona-se apenas o cartão de crédito a liquidar e preenche-se o "Montante" a transferir entre a conta e o cartão de crédito. Este pagamento também pode ser agendado para um determinado dia ou segundo uma periodicidade.

Após o preenchimento dos dados é necessário em todos os pagamentos que o utilizador se autentique com uma matriz e por fim é lhe apresentado o comprovativo do pagamento.

Novo Banco: NB smart app

Uma aplicação que possui alguns fatores de inovação interessantes como o reconhecimento de uma fatura a partir do campo de pagamento por multibanco [4]. As restantes sub-funcionalidades de pagamento são bastante semelhantes às presentes na aplicação descrita anteriormente.

Montepio: M24

Esta aplicação possui uma interface que se destaca. À semelhança da aplicação NB smart app, esta também possui a funcionalidade de reconhecimento de uma fatura [3]. As restantes sub-funcionalidades de pagamento são bastante semelhantes às presentes nas aplicações anteriores.

[1] Os pagamentos de serviços podem ser agendados com uma periodicidade, esta pode ser única, semanal, quinzenal, mensal, trimestral, semestral e anual

Capítulo 3

Análise de Requisitos

Neste capítulo serão analisados os requisitos necessários para o desenvolvimento da aplicação. Nestes encontram-se os principais *stakeholders* da aplicação, os requisitos funcionais e não funcionais, os casos de uso e o planeamento. Pretende-se com este capítulo que o leitor fique com uma ideia mais detalhada acerca dos conteúdos da aplicação.

3.1 *Stakeholders*

Normalmente nas aplicações de *homebanking* os *stakeholders* utilizadores são os clientes que possuem pelo menos uma conta no banco detentor da aplicação. Os requisitos de qualidade mais importantes para os clientes do banco serão o desempenho, a disponibilidade, a fiabilidade, a segurança, a experiência de utilização e a usabilidade.

Apesar de não haver um banco detentor da aplicação em desenvolvimento, estes requisitos serão considerados de qualquer forma. Como único programador da aplicação, constituo um *stakeholder* não utilizador e considerarei como requisitos de qualidade mais importantes a modificabilidade, a testabilidade e a reutilização.

3.2 Requisitos Funcionais

Esta aplicação irá conter duas funcionalidades, Login e Pagamentos, que contêm variadas sub-funcionalidades dentro das mesmas. De seguida serão definidas todas estas funcionalidades e sub-funcionalidades.

- Utilizador não autenticado

  RF1: Login

- Utilizador autenticado

  RF2: Logout

  - Possui matriz de autenticação

    RF3: Pagamento de serviços

    RF4: Pagamento ao estado

    RF5: Pagamento da segurança social

    RF6: Pagamento por QR Code

RF7: Pagamentos frequentes

RF8: Reconhecimento de fatura

RF9: Pagamento agendado

RF10: Notificação de pagamento agendado

RF11: Comprovativo de pagamento

RF12: Autenticação com matriz

RF13: Autenticação com impressão digital

– Não possui matriz de autenticação

RF14: Criar matriz de autenticação

Os requisito funcionais da aplicação móvel são explicados detalhadamente na Secção 3.4.

3.3 Requisitos Não Funcionais

Para garantir a qualidade da aplicação esta deve cumprir alguns requisitos de qualidade, requisitos estes que são apresentados nesta secção.

- **Desempenho:** O tempo de espera do utilizador deve ser minimizado ao máximo.
- **Disponibilidade:** O utilizador deve conseguir ter acesso à aplicação em qualquer instante de tempo.
- **Fiabilidade:** A aplicação deve-se manter sempre operável e com o mínimo de falhas, faltas e erros possível.
- **Segurança:** Este requisito é muito importante em aplicações deste âmbito. Para que seja cumprido, as informações mais sensíveis devem ser encriptadas. Deve também ser cumprido o requisito 2FA explicado na Secção 2.1.6.
- **Experiência de utilização:** Pode-se afirmar, tendo como referência a Secção 2.1.3, que para haver uma continuidade de utilização da aplicação por parte do utilizador, esta terá de fornecer ao mesmo uma boa experiência de utilização. Para que isto seja cumprido a aplicação deverá ter uma interface agradável e simples e, deve falhar o mínimo possível, de maneira a que o utilizador se sinta bem a utilizar a mesma.
- **Usabilidade:** A aplicação deverá ser simples, intuitiva e fácil de utilizar.
- **Modificabilidade:** Caso haja a necessidade de serem feitas alterações na aplicação, este requisito facilita a realização das tarefas correspondentes à respetiva alteração.
- **Testabilidade:** Todos os componentes do sistema devem ser fáceis de testar separadamente e integralmente.
- **Reutilização:** O código da aplicação deve ser reutilizável, de maneira a que seja poupado esforço desnecessário e evitando repetição do código.

3.4 Casos de Uso

Os casos de uso podem ser de dois tipos: simples ou elaborados. Os primeiros consistem numa breve descrição dos casos, enquanto que os segundos contemplam variados pontos. Estes pontos são os seguintes: ator principal; interesses; pré-condições; pós-condições; cenário principal de sucesso; e extensões. Estes casos de uso constituem a informação acerca da utilização da aplicação móvel e são detalhados nas secções seguintes.

Para que se possa ter uma visão geral do uso da aplicação é apresentado um diagrama de casos de uso. Este está presente no anexo A.

3.4.1 UC1 – Efetuar Pagamento de um serviço

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Efetuar o pagamento de um serviço.
- Pré-condições:
  - O utilizador está previamente autenticado;
  - O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.
- Pós-condições:
  - O serviço em causa é pago.
- Cenário principal de sucesso e extensões:
  1. Selecionar a opção “Pagamentos” no menu hambúrguer.
  2. Carregar no botão “Pagamento de Serviços”.
  3. Selecionar a conta a debitar.
  4. Preencher os campos “Entidade”, “Referência” e “Montante”.
     (a) Para o pagamento de telecomunicações, transportes e internet o utilizador pode carregar no botão “Pagamento pré-definido”, seleciona o operador e preencher apenas os campos “Referência” e “Montante”.
  5. (Opcional) Agendar pagamento.
  6. Carregar no botão “Pagar”.
  7. O sistema mostra ao utilizador um *dialog* com os dados do pagamento para que possa ser feita a confirmação do pagamento.
     (a) O sistema informa o utilizador que a entidade, a referência e/ou o montante são inválidos e o caso de uso termina.
  8. O utilizador carrega em “OK”, o caso de uso termina e dá se início ao caso de uso 13.
     (a) O utilizador clica “Cancelar” e o caso de uso termina.

3.4.2 UC2 – Efetuar pagamento ao estado

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Pagar impostos, taxas, custos judiciais, juros de mora, penhoras, entre outros.
- Pré-condições:
  - O utilizador está previamente autenticado;
  - O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.
- Pós-condições:
  - O pagamento ao estado é efetuado.
- Cenário principal de sucesso e extensões:
  1. Selecionar a opção "Pagamentos" no menu hambúrguer.
  2. Carregar no botão "Pagamento ao Estado".
  3. Selecionar a conta a debitar.
  4. Preencher os campos "Referência" e "Montante".
  5. (Opcional) Agendar pagamento.
  6. Carregar no botão "Pagar".
  7. O sistema mostra ao utilizador um *dialog* com os dados do pagamento para que possa ser feita a confirmação do pagamento.
     (a) O sistema informa o utilizador que a referência e/ou o montante são inválidos e o caso de uso termina.
  8. O utilizador carrega em "OK", o caso de uso termina e dá se início ao caso de uso 13.
     (a) O utilizador clica "Cancelar" e o caso de uso termina.

3.4.3 UC3 – Efetuar pagamento da segurança social

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Pagar a segurança social.
- Pré-condições:
  - O utilizador está previamente autenticado;
  - O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.

- Pós-condições:
  - O pagamento à segurança social é efetuado.
- Cenário principal de sucesso e extensões:
  1. Selecionar a opção “Pagamentos” no menu hambúrguer.
  2. Carregar no botão “Pagamento segurança social”.
  3. Selecionar a conta a debitar.
  4. Selecionar o tipo de pagamento.
  5. Selecionar o tipo de remuneração.
  6. Preencher os campos “Número de Beneficiário” e “Remuneração Mensal”.
  7. Selecionar o período de pagamento (Ano/Mês).
  8. O sistema exibe o montante.
  9. (Opcional) Agendar pagamento.
  10. Carregar no botão “Pagar”.
  11. O sistema mostra ao utilizador um *dialog* com os dados do pagamento para que possa ser feita a confirmação do pagamento.
      (a) O sistema informa o utilizador de que o número de beneficiário inserido não é válido e o caso de uso termina.
  12. O utilizador carrega em “OK”, o caso de uso termina e dá se início ao caso de uso 13.
      (a) O utilizador clica “Cancelar” e o caso de uso termina.

3.4.4 UC4 – Reconhecimento de uma fatura a partir de um QR Code

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Efetuar um pagamento sem a necessidade de preencher os campos entidade, referência e montante.
- Pré-condições:
  - O utilizador está previamente autenticado;
  - O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.
- Pós-condições:
  - O utilizador é redirecionado para a página de “Pagamento de Serviços” ou “Pagamento ao Estado” com os campos “Entidade”, “Referência” e “Montante” já preenchidos.
- Cenário principal de sucesso e extensões:

1. Selecionar a opção “Pagamentos” no menu hambúrguer.
2. Carregar no botão “Pagamento QR Code”.
3. Apontar, com a câmara do *smartphone*, para o QR Code presente na fatura.
4. O sistema deteta o QR Code.
5. O utilizador é redirecionado para a página “Pagamento de serviços” com os campos “Entidade”, “Referência” e “Montante” já preenchidos e o caso de uso termina.
   (a) O utilizador é redirecionado para a página “Pagamento ao Estado” com os campos referência e montante já preenchidos e o caso de uso termina.
   (b) O sistema informa o utilizador que o QR Code é inválido e o caso de uso termina.

3.4.5 UC5 – Efetuar um pagamento periódico

- Ator principal:
  – Utilizador que possui matriz de autenticação.
- Interesses:
  – Efetuar o pagamento de um serviço repetidamente no período e na data indicados.
- Pré-condições:
  – O utilizador está previamente autenticado;
  – O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.
- Pós-condições:
  – O serviço em causa é pago repetidamente no período indicado.
  – O utilizador é notificado cada vez que o pagamento é efetuado.
- Cenário principal de sucesso e extensões:
  1. Selecionar a opção “Pagamentos” no menu hambúrguer.
  2. Carregar no botão “Pagamento de Serviços”.
  3. Selecionar a conta a debitar.
  4. Preencher os campos “Entidade”, “Referência” e “Montante”.
  5. Selecionar a periodicidade do pagamento.
  6. Selecionar a data da operação.
  7. Carregar no botão “Pagar”.
  8. O sistema mostra ao utilizador um *dialog* com os dados do pagamento para que possa ser feita a confirmação do pagamento.
     (a) O sistema informa o utilizador que a referência, a entidade e/ou o montante são inválidos e o caso de uso termina.
  9. O utilizador carrega em “OK”, o caso de uso termina e dá se início ao caso de uso 13.
     (a) O utilizador clica “Cancelar” e o caso de uso termina.

3.4.6 UC6 – Selecionar um pagamento frequente

- Ator principal:

  – Utilizador que possui matriz de autenticação.

- Interesses:

  –Facilitar o preenchimento dos campos de um pagamento recorrendo a um pagamento frequente.

- Pré-condições:

  – O utilizador está previamente autenticado;

  – O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.

- Pós-condições:

  – O utilizador é redirecionado para a página “Pagamento de serviços” ou “Pagamento ao estado” com os campos “Entidade”, “Referência” e “Montante” já preenchidos.

- Cenário principal de sucesso e extensões:

  1. Selecionar a opção “Pagamentos” no menu hambúrguer.
  2. Carregar no botão “Pagamento de Serviços”.
     (a) Carregar no botão “Pagamento ao estado”.
  3. Carregar no botão “Frequentes”.
  4. Selecionar a conta a debitar.
  5. Perante a lista de pagamentos frequentes, selecionar um pagamento.
     (a) O sistema informa o utilizador que a conta selecionada não possui pagamentos frequentes.
  6. O utilizador é redirecionado para a página “Pagamento de serviço” com os campos “Entidade”, “Referência” e “Montante” já preenchidos.
     (a) O utilizador é redirecionado para a página “Pagamento ao estado” com os campos “Referência” e “Montante” já preenchidos.

3.4.7 UC7 – Reconhecimento dos dados de pagamento de uma fatura

- Ator principal:

  – Utilizador que possui matriz de autenticação.

- Interesses:

  – Facilitar o preenchimento dos campos de um pagamento a partir do reconhecimento do campo onde estão presentes os dados para o pagamento por multibanco de uma fatura.

- Pré-condições:
  - O utilizador está previamente autenticado;
  - O utilizador possui uma matriz de autenticação, que pode ser criada na aplicação de acordo com o UC12.
- Pós-condições:
  - O utilizador é redirecionado para a página "Pagamento de serviços" ou "Pagamento ao Estado" com os campos "Entidade", "Referência" e "Montante" já preenchidos.
- Cenário principal de sucesso e extensões:
  1. Selecionar a opção "Pagamentos" no menu hambúrguer.
  2. Carregar no botão "Pagamento de Serviços".
     (a) Carregar no botão "Pagamento ao estado".
  3. Carregar no botão "Detetar".
  4. Apontar, com a câmara do *smartphone*, para o campo "Pagar por multibanco" presente na fatura.
  5. A fatura é reconhecida.
  6. O utilizador é redirecionado para a página "Pagamento de Serviços" com os campos "Entidade", "Referência" e "Montante" já preenchidos e o caso de uso termina.
     (a) O utilizador é redirecionado para a página "Pagamento ao estado" com os campos "Referência" e "Montante" já preenchidos.

3.4.8 UC8 – Enviar comprovativo por email

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Enviar o comprovativo de pagamento para um endereço de email indicado.
- Pré-condições:
  - O utilizador realizou um dos casos de uso UC1, UC2 ou UC3 com sucesso.
- Pós-condições:
  - É enviado um email com o comprovativo do pagamento em questão.
- Cenário principal de sucesso e extensões:
  1. Na página "Comprovativo de pagamento", carregar no botão "Enviar por email".
  2. Preencher os campos "Email", "Nome" (opcional) e "Descrição" (opcional).
  3. Carregar no botão "Enviar".
  4. O sistema informa o utilizador que o email foi enviado com sucesso e o caso de uso termina.
     (a) O sistema informa o utilizador que o email é inválido com uma mensagem de erro.

3.4.9 UC9 – Download do comprovativo

Após o utilizador ter efetuado o caso de uso UC1, UC2 ou UC3 com sucesso, na página comprovativo de pagamento, carregar no botão “Download”.

3.4.10 UC10 – Eliminar um pagamento agendado

- Ator principal:
  - Utilizador que possui matriz de autenticação.
- Interesses:
  - Cancelar um pagamento que está agendado.
- Pré-condições:
  - O utilizador possui um ou mais pagamentos agendados.
- Pós-condições:
  - O pagamento que estava agendado foi cancelado.
- Cenário principal de sucesso e extensões:
  1. O utilizador seleciona a opção “Pagamentos” no menu hambúrguer.
  2. Carrega no botão “Pagamentos Agendados”.
  3. Na lista de pagamentos agendados, pressionar o pagamento que deseja cancelar até que a linha mude de cor.
     (a) Na lista de pagamentos agendados, pressionar o pagamento que deseja cancelar até que a linha mude de cor e, de seguida, seleciona mais do que um pagamento clicando apenas uma vez na posição da lista desejada.
  4. Carregar no ícone da *toolbar* que representa um caixote do lixo.
  5. O sistema elimina os pagamentos selecionados.
     (a) O sistema informa o utilizador de que não foi possível cancelar os pagamentos selecionados e o caso de uso termina.
  6. A lista de pagamentos agendados é atualizada e o caso de uso termina.

3.4.11 UC11 – Notificação de pagamento

No dia em que é realizado um pagamento que foi agendado é enviada uma notificação a avisar o utilizador que o pagamento foi efetuado. Quando este abre a notificação, a página de Login é apresentada e, caso se autentique com sucesso, é redirecionado para a página “Comprovativo de pagamento”.

3.4.12 UC12 – Criar matriz de autenticação

- Ator principal:
  - Utilizador sem matriz de autenticação.

- Interesses:
  - Criar uma matriz de autenticação para que seja possível efetuar pagamentos.
- Pré-condições:
  - O utilizador está previamente autenticado.
- Pós-condições:
  - É enviada uma matriz de autenticação para o email do utilizador e este passa a poder utilizar todas as funcionalidades de pagamentos.
- Cenário principal de sucesso e extensões:
  1. O utilizador seleciona a opção "Pagamentos" no menu hambúrguer.
  2. O sistema mostra um *dialog* a informar que o utilizador não pode efetuar esta operação.
  3. Carregar no botão "Criar matriz".
     (a) Carregar no botão "Cancelar" e o caso de uso termina.
  4. Uma matriz de autenticação é enviada para o email do utilizador e o caso de uso termina.
     (a) O sistema informa o utilizador que ocorreu um erro e o caso de uso termina

3.4.13 UC13 – Autenticação do pagamento

- Ator principal:
  - Utilizador com matriz de autenticação.
- Interesses:
  - Autenticar-se de maneira a poder efetuar pagamentos.
- Pré-condições:
  - O utilizador está previamente autenticado.
  - O utilizador confirmou a realização de um pagamento.
  - O utilizador realizou um dos casos de uso UC1, UC2 ou UC3 com sucesso.
- Pós-condições:
  - O pagamento é efetuado.
- Cenário principal de sucesso e extensões:
  1. Após a realização da confirmação do pagamento, o sistema irá mostrar ao utilizador um *dialog* para que seja realizada a autenticação.
  2. O utilizador coloca o dedo no leitor de impressões digitais.
     (a) O utilizador carrega no botão "Autenticar com matriz" e preenche as posições da matriz pedidas, carregando de seguida no botão "OK".
     (b) O utilizador preenche as posições da matriz pedidas, carregando de seguida no botão "OK".

3. O sistema informa o utilizador que foi autenticado com sucesso.
   (a) O sistema mostra um *dialog* ao utilizador com a informação de que ocorreu um erro na autenticação

3.4.14 UC14 – Login

- Ator principal:
  – Utilizador não autenticado.
- Interesses:
  – Efetuar autenticação perante a aplicação, podendo assim usufruir das funcionalidades da mesma.
- Pré-condições:
  – Utilizador necessita de possuir uma conta HomeBanking.
- Pós-condições:
  – O utilizador é autenticado.
- Cenário principal de sucesso e extensões:
  1. Abrir a aplicação HomeBanking.
  2. Preencher os campos “Número de Contrato” e “Código de Acesso”.
  3. Carregar no botão “Entrar”.
  4. O sistema redireciona o utilizador para a página principal e o caso de uso termina.
     (a) O sistema informa o utilizador que o número de contrato e/ou o código de acesso estão errados e o caso de uso termina.

3.4.15 UC15 – Logout

O utilizador indica que quer efetuar Logout e é redirecionado para página de login, deixando, assim, de estar autenticado pelo sistema.

3.5 Planeamento

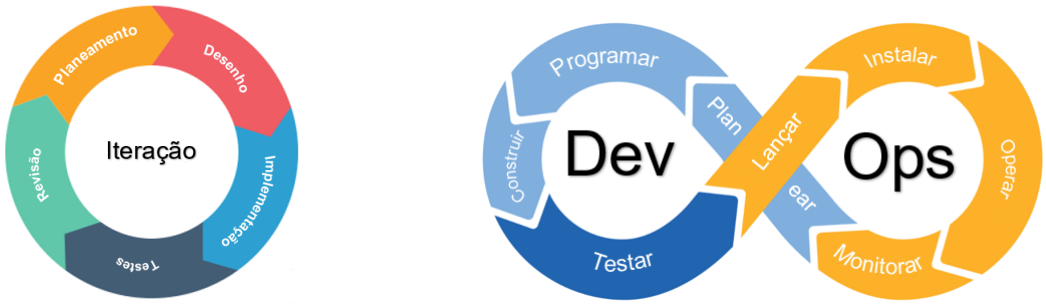
Nesta secção serão descritos os recursos utilizados, assim como, o planeamento do projeto e a metodologia utilizada no decurso do mesmo.

3.5.1 Metodologia

As empresas atualmente utilizam maioritariamente dois tipos de metodologia de trabalho. A primeira é a *Agile*, uma metodologia caracterizada pelo processamento iterativo, realizando em primeiro lugar o planeamento, de seguida o desenho da solução, depois a implementação e testes, e por fim, é feita a revisão das alterações necessárias (Figura 3.1a). A segunda metodologia é a *DevOps*, esta une dois

conceitos: o desenvolvimento e as operações (Figura 3.1b). Esta metodologia é mais virada para o meio empresarial com a intenção de interligar os departamentos de desenvolvimento e operações.

A metodologia utilizada no decurso deste projeto foi a metodologia *Agile*, sendo que cada funcionalidade e sub-funcionalidade desenvolvidas para a aplicação representam uma iteração. Para cada iteração realizou-se os processos referidos acima. Após a realização de todas as iterações foram concretizados testes de utilização.



(a) Metodologia Agile

(b) Metodologia DevOps

Figura 3.1: Esquema dos dois tipos de metodologias de trabalho

3.5.2 Recursos

Hardware

- Smartphone Lenovo K5
  - Sistema Operativo: Android™ 5.1, Lollipop
  - Processador: Octa-core 1.4 GHz
  - Memória RAM: 2GB
  - Câmara: 13MP
  - Resolução de Ecrã: 5.0”(1280 x 720)
- Laptop Dell Latitude E5250
  - Sistema Operativo: Windows 10 Pro
  - Processador: Intel Core i5 2 GHz
  - Memória RAM: 8 GB
  - Disco Rígido: 128 GB SSD
  - Placa Gráfica: Intel HD Graphics 4400
  - Resolução de Ecrã: 5.0”(1280 x 720)

Software

- Microsoft Windows 10 64-bit
- Android Studio 3.1 (Desenvolvimento da aplicação móvel)

- Eclipse Oxygen 3a (Desenvolvimento do web server)
- Microsoft SQL Server Management Studio 17 (Desenvolvimento da base de dados)
- Vectr (Desenho do logótipo da aplicação)
- Pixlr (Desenho dos cartões de crédito, débito e gold)

Emulador

Foram utilizados dois emuladores da ferramenta Android Virtual Device presente no Android Studio. A única diferença entre estes dois emuladores é o sistema operativo.

- Nexus 5X API 27
  - Sistema Operativo: Android™ 8.1, Oreo
  - Processador: Dual-core
  - Memória RAM: 1536MB
  - Resolução de Ecrã: 5.2”(1080 x 1920)
- Nexus 5X API 16
  - Sistema Operativo: Android™ 4.1, Jelly Bean
  - Processador: Dual-core
  - Memória RAM: 1536MB
  - Resolução de Ecrã: 5.2”(1080 x 1920)

3.5.3 Plano de Trabalhos

2 de Novembro e 2017 – 31 de Dezembro de 2017:

- Levantamento de requisitos
- Análise funcional, técnica e arquitetural
- Desenho funcional

1 de Janeiro de 2018 – 31 de Janeiro de 2018:

- Desenvolvimento de protótipos

1 de Fevereiro de 2018 – 31 de Maio de 2018:

- Implementação técnica da solução
- Configuração da solução integrada
- Realização de testes integrados

1 de Junho de 2018 – 31 de Junho de 2018:

- Elaboração do relatório final

Alterações do plano de trabalhos

O foco inicial do projeto consistia em desenvolver uma aplicação móvel *homebanking* no âmbito empresas, para um cliente na área da banca. Devido a alguns imprevistos, a aplicação móvel deixou de ser destinada ao respetivo cliente. Apesar disso, o objetivo do projeto manteve-se, mas devido a este percalço houve um pequeno atraso na fase de implementação técnica da solução e nas seguintes fases, consequentemente. Também não será possível a exibição dos protótipos devido a estes terem informações do cliente.

Além disto, como houve liberdade de decisão ao longo de todo o desenvolvimento, após a fase de implementação técnica da solução, que foi feita na linguagem *Java*, optou-se por fazer a migração de toda a aplicação para a linguagem *Kotlin*.

Capítulo 4

Desenho da Solução

4.1 Arquitetura do Sistema

Para este sistema foi escolhida a arquitetura SOA (*Service Oriented Architecture*) que, como o nome diz, é uma arquitetura orientada a serviços. Esta é caracterizada por estruturar sistemas que podem ser organizados em torno de um número independente de consumidores e fornecedores de serviços distribuídos muito possivelmente por diferentes organizações [21].

Na Figura 4.1 é representada a arquitetura do sistema através de uma vista de componentes e conectores. Nesta é possível ver os vários componentes que constituem o sistema e como estes se conectam. No canto inferior direito, da figura, encontra-se uma legenda que identifica os tipos de conexão.

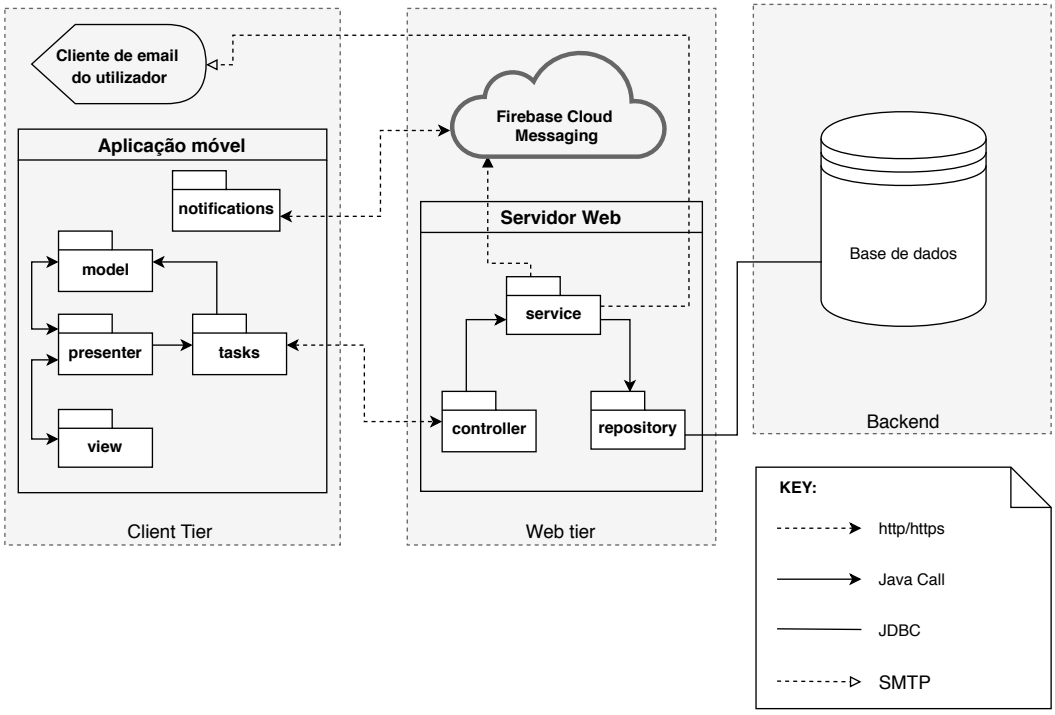


Figura 4.1: Modelo da arquitetura do sistema

Esta arquitetura é constituída apenas por dois consumidores (aplicação móvel e cliente de email do utilizador) e dois fornecedores (Servidor Web e FCM). Por parte do servidor são fornecidos vários serviços que serão consumidos tanto pela aplicação móvel, como pelo cliente de email do utilizador. O

primeiro consome os serviços a partir de um conector REST, que efetua uma comunicação *request/reply* suportada pelo HTTP. Já o segundo consome serviços a partir de emails que são enviados por parte do servidor, a partir de um conector SMTP.

O servidor comunica com a base de dados a partir de um conector JDBC, de maneira a persistir e adquirir as informações necessárias. Para a receção e o envio de *push notifications*, a aplicação móvel e o servidor comunicam com o FCM a partir de pedidos HTTP. A aplicação móvel possui um pacote denominado *"notifications"* que contém classes responsáveis por efetuar o registo do dispositivo e apresentar notificações ao utilizador. Este não apresenta conexões com outros pacotes da aplicação móvel, pois guarda as informações que outros pacotes necessitaram nas SharedPreferences. Para além disso, as classes nele contidas são apenas chamadas a partir de serviços do FCM. Dentro do servidor e da aplicação móvel, são feitas chamadas Java para a comunicação entre pacotes.

4.2 Padrão arquitetural

O padrão arquitetural que será aplicado é o Model-View-Presenter, esta decisão foi efetuada após uma pesquisa onde se pôde comparar os padrões mencionados anteriormente [11].

O padrão Model-View-Controller é indicado para aplicações mais simples que tenham, por exemplo, apenas dois ecrãs. Isto porque, este padrão não é tão complexo como os restantes dois. Para aplicações de maiores dimensões existem padrões mais indicados, sendo estes o padrão Model-View-ViewModel (MVVM) e o Model-View-Presenter (MVP). Estes tornam a aplicação mais modular, testável e de fácil manutenção. A diferença entre estes dois padrões foca-se na ligação entre a *View* e o *Presenter/ViewModel*, sendo que a ligação com a camada *Model* se mantém igual para os dois padrões. Esta diferença é representada na Figura 4.2. No padrão MVVM a camada *ViewModel* não possui nenhuma referência da *View*, possuindo apenas os dados que serão utilizados pela *View*. Já no padrão MVP as camadas *Presenter* e *View* estão interligadas por uma interface contrato e, por isso, possuem referências uma da outra [26]. Apesar do padrão MVVM estar a ser cada vez mais utilizado e, no que diz respeito à programação assíncrona ser mais eficaz e fácil de perceber, a escolha pelo outro padrão, deveu-se à simplicidade da lógica de desenvolvimento e da limpeza do código, que se considera como pontos a favor deste padrão. A Google aconselha a que se utilizem os dois padrões, o *Presenter* para programação síncrona e o *ViewModel* para programação assíncrona. Na implementação do padrão MVP segui-se a amostra *todo-mvp*, que consiste num projeto criado pela comunidade *Android*, com a ajuda da Google [5].

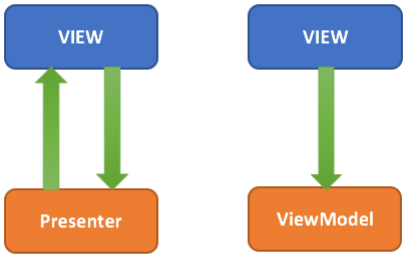


Figura 4.2: Diferenças entre o padrão MVP e o padrão MVVM

Na prática, haverá uma interface *Contract*, por exemplo *LoginContract*, que irá conter duas interfaces, a *View* e a *Presenter*. A interface *View* contempla os métodos que serão utilizados pelo *Presenter* e, vice-versa. As classes que implementam as interfaces *View* são responsáveis por tratar da interação com o utilizador e têm como função a exibição de informação e o reconhecimento das ações realizadas pelo utilizador. Estas classes são *Activitys* ou *Fragments*. As classes que implementam as interfaces *Presenter* têm como função o tratamento das ações do utilizador, a atualização da interface e a comunicação com o servidor web. Para facilitar a realização de testes, estas classes não devem estar dependentes da *View*, de forma a que se possam fazer testes à classe *Presenter* sem que seja necessário fazer *mocks* da classe *View*.

4.3 Modelo da Base de Dados

Este projeto é composto por uma base de dados constituída por apenas 10 tabelas, que serão apresentadas nas secções seguintes. O modelo de base de dados é representado no Anexo B.

4.3.1 Client

Esta tabela representa um cliente de um banco, e possui as seguintes informações:

id: chave primária que representa o número de cliente. Este contém 9 dígitos e é gerado automaticamente;

cc: número de identificação;

nif: número de identificação fiscal;

first_name/last_name: primeiro e último nome, respetivamente;

phone: contacto telefónico;

email: endereço de correio eletrónico.

4.3.2 Account

Representa uma conta bancária e contém as seguintes informações:

id: chave primária que representa o número de conta. Este contém 9 dígitos e é gerado automaticamente;

client_id: chave estrangeira que representa o número de cliente detentor da conta. Esta referencia a chave primária da tabela Client;

type: tipo de conta. Informa se a conta é de débito, crédito entre outras;

password: palavra-passe representada por 4 dígitos. Esta simboliza os números secretos pedidos pela máquina de multibanco, sendo meramente simbólica e não utilizada nesta aplicação;

current_balance: saldo disponível, que como o nome indica, é o saldo que o cliente possui na conta;

BIC_SWIFT: código de identificação do banco. É informação de carácter simbólico, da mesma forma que a password.

4.3.3 User

Representa o utilizador da aplicação e contém as seguintes informações:

id: chave primária que representa o número de contrato. Esta contém 8 dígitos que identificam o utilizador, sendo estes gerados automaticamente quando o utilizador se regista na aplicação homebanking;

password: palavra-passe composta por 6 dígitos. Esta é encriptada pelo servidor web e, posteriormente, persistida na base de dados;

client_id: chave estrangeira que representa o número de cliente. Esta referencia o cliente que se registou na aplicação homebanking, associando todas as contas do mesmo à aplicação;

active/role: estas colunas são necessárias por motivos de configurações de segurança do servidor web que para autenticar um utilizador obriga a ter estes dois valores. A coluna ”active”assume o valor 1 quando a conta está ativa e 0 caso contrário. Já a coluna ”role”assume sempre o valor ”USER”;

matrix_id: chave estrangeira que representa a matriz de autenticação do utilizador. Esta referencia a tabela Matrix que será pormenorizada na Secção 4.3.9;

token_notifications: token do dispositivo que serve para identificar o dispositivo, de modo a que este receba notificações. Este tema é explicado em mais detalha no próximo capítulo;

tries: as tentativas de login mal sucedidas são contabilizadas e guardadas na coluna tries, servindo para reforçar a segurança. A conta é bloqueada quando se atinge um certo número de tentativas consecutivas.

4.3.4 Payment

Representa um pagamento e contém os seguintes atributos:

id: chave primária que representa a identificação do pagamento;

from_account: chave estrangeira que representa a conta na qual foi debitado o valor do pagamento. Esta referencia a tabela ”Account”;

entity: chave estrangeira que representa a entidade do pagamento. Esta referencia a tabela ”Company”;

amount: montante do pagamento. É o valor do pagamento realizado pelo utilizador;

date: data do pagamento. É a data em que o pagamento foi ou será realizado;

reference: chave estrangeira que representa a referência do pagamento. Esta referencia a tabela ”Bill”.

context: tipo do pagamento. Pode tomar os valores ”serv”, ”imp”e ”ss”, definindo assim se este é um pagamento de serviços, impostos ou segurança social, respetivamente.

4.3.5 Periodic_Payment

Esta tabela representa os pagamentos agendados para serem realizados periodicamente e, contém as seguintes informações:

id: chave primária que representa a identificação do pagamento periódico. Esta serve para distinguir os diferentes pagamentos periódicos;

payment_id: chave estrangeira que representa a identificação do pagamento. Esta referencia a tabela "Payment";

periodicity: periodicidade do pagamento, representando o período de espera até ao próximo pagamento. Este pode tomar os valores "UNIC", "WEEKLY", "BIWEEKLY", "MONTHLY", "QUARTERLY", "SEMIANNUALY" e "ANNUALY" para pagamentos únicos, semanais, quinzenais, mensais, trimestrais, semestrais e anuais, respetivamente;

date: data em que o pagamento foi ou será realizado;

user_id: chave estrangeira que representa um utilizador. Esta referencia a tabela "User" e é utilizada para o envio de um email e notificação quando um pagamento deste tipo é efetuado.

4.3.6 SS_Payment

O pagamento da segurança social possui algumas características diferentes dos restantes tipos de pagamento e, por este motivo, é necessária esta tabela que é uma entidade fraca da tabela Payment. Esta contém as seguintes informações:

payment_id: chave estrangeira que representa a identificação do pagamento. Esta referencia a tabela "Payment", ou seja, a identificação do pagamento em questão;

ss_number: representa o NISS e serve para identificar a quem é atribuído o pagamento;

payment_type: tipo de pagamento. Este pode tomar os valores "Trabalhadores por conta de outrem", "Trabalhadores independentes" e "Seguro social voluntário";

remuneration_type: tipo de remuneração. Este pode tomar os valores "Mês completo", "Mês incompleto", "Horária";

month/year: mês e ano, respetivamente, que representam o período a que o pagamento se refere.

time: horas ou dias. Quando o tipo de remuneração é "Mês incompleto" ou "Horária", o utilizador necessita de colocar o número de dias ou o número de horas trabalhadas respetivamente. Esses valores serão persistidos nesta coluna.

4.3.7 Company

Esta tabela representa uma entidade, podendo ser uma empresa, uma organização, o estado, entre outros. Esta contém seguintes informações:

entity: chave primária que representa a entidade. Esta é composta por 5 dígitos que identificam a empresa, organização, estado, entre outros;

name: nome da entidade;

category: categoria da entidade. Esta pode ser, por exemplo, transportes, telecomunicações, entre outros.

balance: saldo da entidade. Serve apenas para verificar que a entidade recebe o valor do montante de um pagamento, na simulação.

4.3.8 Bill

Esta tabela representa uma fatura e contém as seguintes informações:

reference: chave primária que representa a referência da fatura. Esta contém 9 dígitos, no caso de ser uma fatura de um serviço e, 15 dígitos, no caso de ser uma fatura do estado. Estes dígitos identificam a fatura;

company_id: chave estrangeira que representa a entidade da fatura. Esta referencia a tabela "Company";

client_id: chave estrangeira que representa o número do cliente. Esta referencia a tabela "Client"e serve para que a fatura possa ser enviada por email ao respetivo cliente;

amount: montante que é mostrado na fatura;

state: estado do pagamento. Este toma o valor "NOT_PAYED" quando ainda não foi pago e "PAYED" após ter sido pago;

description: descrição do produto/serviço/imposto.

4.3.9 Matrix

Esta tabela representa uma matriz de autenticação. Esta apenas contém o id da matriz, mas é referenciada por 25 células (Cell).

4.3.10 Cell

Esta tabela é uma entidade fraca da tabela "Matrix" e representa uma célula de uma respetiva matriz. Esta contém as seguintes informações:

matrix_id: chave estrangeira que serve para identificar a matriz à qual esta célula se refere. Esta referencia a tabela "Matrix".

row: linha da célula. Representa o valor da posição horizontal da célula na matriz;

col: coluna da célula. Representa o valor da posição vertical da célula na matriz;

val: valor da célula. Este é representado por um dígito, sendo este o valor da posição na matriz.

4.4 Navegação

Existem padrões de design estipulados pela Google para que a utilização das aplicações seja facilitada devido à familiarização dos utilizadores com o modo de navegação da mesma. Por exemplo, a utilização de

um menu hambúrguer. A navegação numa aplicação deve ser o mais simples possível, sendo que quantos menos passos o utilizador precisar de fazer para chegar ao destino pretendido, melhor é a navegação. Na Figura 4.3 é representado, em esquema, a navegação da aplicação. Como se pode ver, pela figura, para efetuar qualquer pagamento basta efetuar 3 passos: o login; o menu hambúrguer; e o menu de pagamentos.

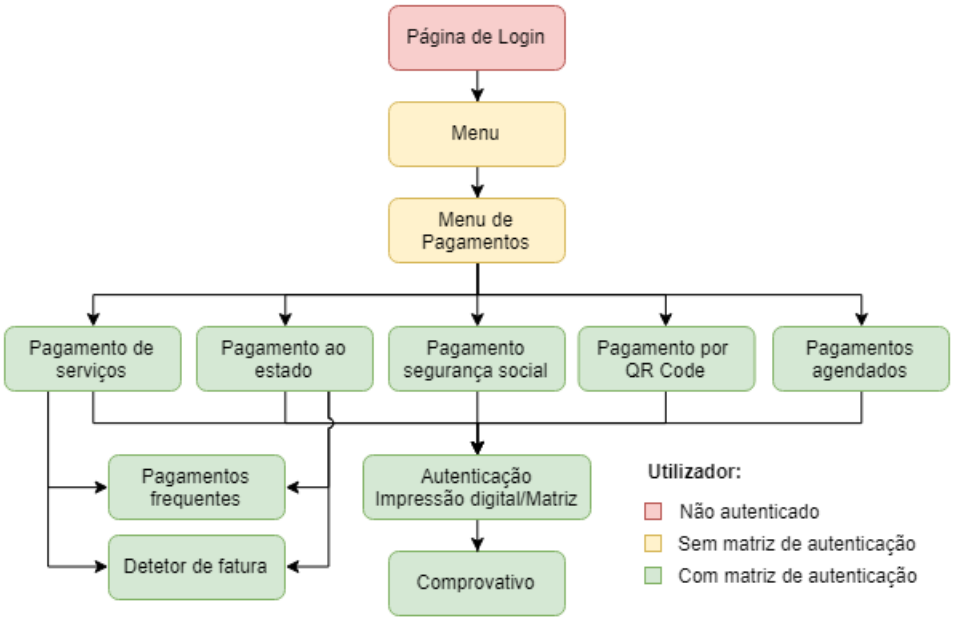


Figura 4.3: Navegação na aplicação

Capítulo 5

Implementação

Neste capítulo são apresentadas as várias etapas do processo de implementação da solução em desenvolvimento neste projeto. O processo é repartido por três partes: aplicação móvel; servidor web; e base de dados. Nas secções que se seguem será descrito o desenvolvimento dos mesmos, dando especial ênfase à aplicação móvel.

5.1 Aplicação Android

Nesta secção será explicada a estrutura da aplicação, a implementação de todas as funcionalidades e a migração, da mesma, da linguagem Java para Kotlin.

Esta aplicação foi desenvolvida com a ajuda da IDE Android Studio, ferramenta oficial para o desenvolvimento de aplicações nativas para o sistema operativo Android. As linguagens utilizadas foram Java, para a programação do *backend*, e XML para a programação do *frontend*. Para a automação da construção do projeto é utilizada a ferramenta Gradle.

5.1.1 Estrutura

A aplicação segue a estrutura do *template* do Android Studio, separando o Manifest pela pasta "manifest", o *backend* pela pasta "java", o *frontend* pela pasta "res" e a construção automática pela pasta "Gradle Scripts". Esta estrutura é apresentada na Figura 5.1.

A pasta "manifest" contém apenas um ficheiro XML responsável pelas configurações e permissões da aplicação.

O *backend* é estruturado pelos pacotes "home", "login", "notifications", "payments", "offline" e "transactionAuth", que representam as funcionalidades da aplicação. Para além das funcionalidades, no *backend* tem-se ainda os seguintes pacotes: "dialogs", que contém os *AlertDialogs* de sucesso e de erro que são utilizados por toda a aplicação; "models" que corresponde à camada Model do padrão de arquitetura utilizado, contendo as classes responsáveis por gerir os dados; "util" que corresponde às utilidades, como por exemplo um conversor de *doubles* para *strings* com a representação da moeda.

No *frontend* estão presentes alguns pacotes responsáveis pela interface da aplicação, como: "drawble", onde estão presentes as imagens apresentadas na aplicação; "layout", onde se encontram os ficheiros XML com o código dos ecrãs da aplicação; e, por exemplo, o pacote "values", composto por vários ficheiros XML responsáveis pelos idiomas, estilos, cores e dimensões da aplicação.

A estrutura da aplicação contém ainda a pasta ”Gradle Scripts”onde estão presentes os ficheiros de construção automática do projeto. Estes ficheiros possuem configurações, propriedades e importações de bibliotecas externas.

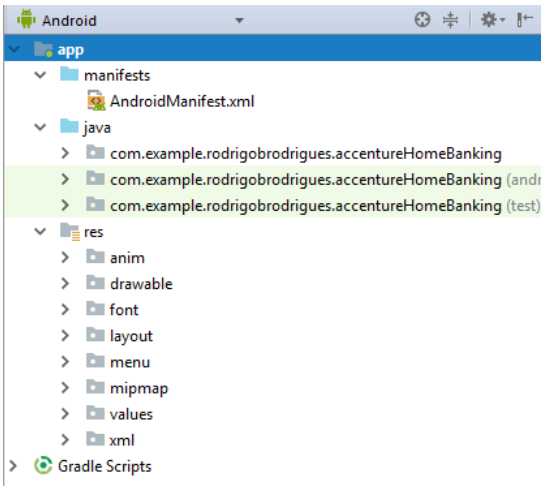


Figura 5.1: Estrutura da aplicação Android

5.1.2 Funcionalidades

Nesta secção são apresentadas todas as funcionalidades presentes na aplicação de forma separada.

Autenticação

Para que a aplicação cumpra os requisitos 2FA definidos na Secção 2.1.6, é necessário que haja no mínimo dois tipos de autenticação. Para a aplicação optou-se por três processos de autenticação distintos: o login; a matriz de autenticação; e a impressão digital do utilizador. O primeiro processo será utilizado no acesso à aplicação, enquanto que, os outros dois serão utilizados na realização de pagamentos. Os processos mencionados anteriormente serão detalhados de seguida.

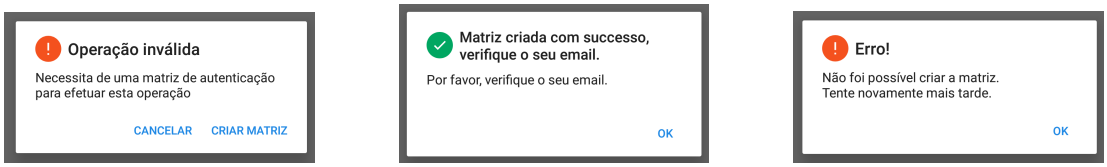
Login:
Esta funcionalidade tem como objetivo autenticar um utilizador, para isso, este deve possuir um número de contrato e um código de acesso. Estes dois dados são geralmente obtidos através de um registo numa caixa de multibanco ou numa agência do banco. Neste caso, será o servidor a fazer a simulação do registo criando e persistindo na base de dados os clientes, as suas contas bancárias e os utilizadores correspondentes a cada cliente. O código de acesso do utilizador é encriptado utilizando uma funcionalidade do *spring security* que tira partido da função de *hashing BCrypt* para efetuar a encriptação do mesmo.

A aplicação tem como página inicial a página de login (Figura C.1a) e, para que o utilizador possa usufruir das restantes funcionalidades deve primeiro autenticar-se nesta página. Para isso basta colocar o número de adesão e o seu código de acesso nos campos pedidos. O utilizador também pode carregar na *checkbox* “Lembrar-me” para que o seu número de adesão seja guardado nas *SharedPreferences* da aplicação. Quando o utilizador carrega no botão para entrar, a aplicação vai comunicar com o *web server*, enviando de forma segura (HTTPS) as credenciais do utilizador. O

servidor por sua vez, a partir de funcionalidades do *spring security*, irá verificar na base de dados se as credenciais colocadas são válidas. Por fim, caso este se tenha autenticado com sucesso, envia os dados do utilizador para a aplicação como objeto JSON, caso contrário, envia uma mensagem de erro.

Matriz de autenticação:

A matriz de autenticação é um processo utilizado para autenticar um pagamento, sendo que sem esta não é possível realizar o mesmo. Desta forma, quando o utilizador carrega na opção "Pagamentos" do menu hambúrguer e não possui uma matriz de autenticação é apresentado um *AlertDialog* informando o utilizador da impossibilidade de efetuar esta operação (Figura 5.2a). Neste é também apresentada a opção de criar uma matriz de autenticação, de modo a realizar todas as operações de pagamento pretendidas. Caso o utilizador clique no botão "CANCELAR" o *AlertDialog* é dispersado, caso o utilizador clique no botão "CRIAR MATRIZ" a aplicação irá fazer um pedido ao servidor para que este crie a matriz. No servidor é criada a matriz, persistindo-a também na base de dados. De seguida, é enviada, para o endereço de correio eletrónico do utilizador, a matriz criada. Este email pode ser visualizado na Figura D.1, em anexo. Quando esta operação é realizada com sucesso, é apresentado um *AlertDialog* com uma mensagem de sucesso (Figura 5.2b), dando ao utilizador o poder de utilizar a funcionalidade de pagamentos. Em caso contrário, é apresentado um *AlertDialog* com uma mensagem de erro (Figura 5.2c).



(a) Dialog para a criação da matriz (b) Sucesso na criação da matriz (c) Erro na criação da matriz

Figura 5.2: AlertDialogs antes e após a criação de uma matriz de autenticação

No processo de autorização de um pagamento, esta matriz é utilizada após um pagamento ser confirmado, pedindo quatro posições aleatórias da mesma. As posições são pedidas ao utilizador através de uma letra e um número inteiro, sendo que a letra refere-se à linha e o número à coluna. Como no código as posições da matriz são representadas por números, foi necessário fazer uma conversão para que a partir de um número fosse obtida uma letra de "A" a "E". Esta conversão é efetuada através do código ASCII. Assim, sabendo que o código ASCII da letra "A" é 65, somando 65 ao número inteiro da linha é possível obter todas as letras referentes à posição linear da matriz. As posições da matriz são pedidas através de um *AlertDialog*, que pode ser visualizado na Figura C.3b, em anexo. Este contém as posições da matriz e por baixo de cada posição contém um *EditText* para que o utilizador insira o correspondente valor da matriz. Quando um valor é inserido, o cursor passa automaticamente para o próximo *EditText*. Para evitar ataques de força bruta, quando o utilizador preenche incorretamente os valores da matriz, o sistema apresenta uma mensagem de erro e volta a pedir 4 novas posições aleatórias.

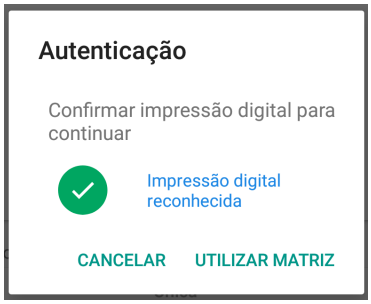
Impressão digital:

Este processo de autenticação é também utilizado para confirmar um pagamento e requer alguns pré-requisitos, tais como:

- uma matriz de autenticação;
- um dispositivo com um sistema operativo Android 6.0 ou superior;
- um leitor de impressões digitais e a impressão digital previamente configurada.

Caso o utilizador não possua um dispositivo com algum dos dois últimos pré-requisitos, então só poderá autenticar o pagamento através da matriz de autenticação. Após a confirmação de um pagamento, é apresentado um *AlertDialog* pedindo ao utilizador para colocar o dedo no leitor da impressão digital. Neste são também apresentados dois botões: o botão "CANCELAR", que faz dispersar o *AlertDialog*; e no botão "UTILIZAR MATRIZ", para realizar o processo através da matriz de autenticação, como é explicado na secção anterior. Este *AlertDialog* pode ser visto na Figura C.3b, em anexo. Quando o utilizador coloca o dedo no leitor de impressões digitais, se a sua impressão digital corresponder à registada no dispositivo, o mesmo será notificado com uma mensagem de sucesso (Figura 5.3a), caso contrário, é apresentada uma mensagem de erro (Figura 5.3b).

Para a implementação desta funcionalidade foi utilizada a ajuda de amostras da Google disponibilizadas no *github* [6] e também da biblioteca *android.hardware.fingerprint* [1].



(a) Sucesso na autenticação por impressão digital

(b) Erro na autenticação por impressão digital

Figura 5.3: AlertDialog com resultado da autenticação por impressão digital

Ambas as mensagens presentes na Figura anterior, são apresentadas apenas durante alguns segundos, desaparecendo automaticamente. Após a mensagem de sucesso da Figura 5.3a desaparecer, o utilizador é informado sobre sucesso ou insucesso do pagamento, através de um *AlertDialog*. Já, em relação à mensagem de erro da Figura 5.3b, quando esta desaparece, o utilizador é retomado para o ecrã da Figura C.3b, em anexo, onde é pedido, novamente, para que este coloque o dedo no leitor de impressão digital.

Página inicial

Numa aplicação completa de *homebanking*, na página inicial da aplicação, são apresentadas geralmente informações como o saldo da conta, os movimentos da mesma, ou até mesmo, publicidade do respetivo banco. Como o objetivo deste projeto é apenas o desenvolvimento da funcionalidade pagamentos, esta página é apresentada em branco, contendo apenas um menu hambúrguer. Neste menu são apresentadas

várias opções, mas apenas a opção "Pagamentos" e a opção "Sair" podem ser selecionadas, sendo que ao clicar nas restantes nada acontece. Este menu é apresentado na Figura C.1c, em anexo.

Menu de pagamentos

Quando o utilizador clica em "Pagamentos" é direcionado para o menu de pagamentos (Figura C.1d). Este menu é constituído por 5 botões dispostos em grelha, sendo que cada um contém um ícone e uma descrição. Os ícones do botão "Pagamento ao Estado" e "Pagamento Segurança Social" foram criados recorrendo à ferramenta Pixlr.

Vista *Offline*

Para este tipo de aplicações é necessário que haja uma conexão à Internet por motivos de segurança e atualização de dados. Por este motivo, quando o utilizador não tem acesso à rede, não é capaz de utilizar as funcionalidades da mesma. Desta forma, foi implementada uma vista que informa o utilizador da necessidade de conexão à Internet caso esta não exista. Quando esta conexão é reposta, o utilizador é redirecionado para a página onde se encontrava anteriormente. Este processo foi implementado utilizando um BroadCastReceiver que faz com que uma função seja chamada sempre que há alterações na rede do *smartphone*. Esta função verifica se o utilizador está ligado à rede ou não. Em caso negativo é mostrado o ecrã apresentado na Figura C.1b, composto por um texto que informa o utilizador de que tem de estar conectado à Internet e por um ícone com *PulseView*. A *PulseView* é uma animação que gera um efeito de pulsação em torno do ícone e é disponibilizada por uma biblioteca externa que pode ser encontrada no *github* [23]. Quando o utilizador volta a ter acesso à Internet a atividade onde estava anteriormente é chamada.

Pagamento de Serviços

Esta funcionalidade tem como finalidade o pagamento de um serviço a partir de uma entidade, referência e montante.

O ecrã é composto por dois botões no topo, destinados à facilitação do preenchimento dos dados. O primeiro encaminha o utilizador para a funcionalidade de pagamentos frequentes e o segundo encaminha o utilizador para a funcionalidade de deteção de uma fatura. De seguida é apresentado um fragmento destinado à seleção da conta de onde será feito o débito do montante do pagamento e um botão destinado a pagamentos de serviços pré-definidos. Todas estas funcionalidades anteriores serão explicadas mais à frente. De seguida, é apresentado um *CardView* destinado ao preenchimento de todas as informações relativas à fatura de pagamento. Este contém um *EditText* para a entidade, três *EditText* para a referência e dois *EditText* para o montante. Quando o utilizador preenche todos os dígitos necessários em cada *EditText* o cursor muda automaticamente para o próximo *EditText*, de forma a melhorar a experiência de utilização. Por fim, apresenta-se um botão para selecionar a periodicidade do pagamento, um fragmento para selecionar a data de pagamento, aparecendo, por defeito, o dia em que este está a ser realizado, e, por último, um botão para continuar a operação. Este ecrã é apresentado nas Figuras C.1e e C.1f, em anexo.

Quando o utilizador carrega no botão "CONTINUAR", em primeiro lugar é verificado do lado do cliente se os campos "Entidade", "Referência" e "Montante" estão devidamente preenchidos. Para isto,

a entidade deverá ter 5 dígitos, a referência, 9 dígitos e, o montante, deve ser superior a 0,00 euros. Quando a validação anterior é efetuada com sucesso, a aplicação faz um pedido ao servidor para validar o conteúdo do pagamento. O servidor por sua vez verifica se existe uma fatura com a entidade e referência indicadas pelo utilizador, verificando também se o saldo da conta a debitar é suficiente para efetuar o pagamento. Em caso positivo é apresentado um *AlertDialog* ao utilizador com os dados do pagamento pedindo para que estes sejam confirmados. Em caso negativo é apresentada uma mensagem de erro. Quando o utilizador confirma o pagamento e se autentica com a matriz de autenticação ou a impressão digital, a aplicação envia para o servidor a informação do pagamento. Este recorre ao serviço de pagamentos, para persistir o pagamento na base de dados, subtrair ao saldo do utilizador o montante do pagamento e somar o mesmo ao saldo da entidade. Por fim o servidor responde à aplicação com o id do pagamento, de modo a que seja possível emitir um comprovativo do pagamento. Este processo de obtenção de um comprovativo será explicado mais à frente.

Pagamento de serviços pré-definidos

Esta opção de pagamento é selecionada dentro do ecrã dos "Pagamentos de Serviços" e é utilizada para preencher automaticamente o campo "Entidade", quando esta corresponde a um serviço de telecomunicações, Internet ou transportes.

Quando o utilizador clica no botão "Serviço pré-definido" são lhe apresentadas opções para a escolha da categoria do serviço, assim como, da operadora que presta esse serviço. Quando as duas opções anteriores são selecionadas, o campo "Entidade" é preenchido automaticamente.

Na Figura 5.4 são apresentados os campos que são adicionados ao ecrã dos "Pagamentos de Serviços" quando se seleciona o campo "Serviço pré-definido".

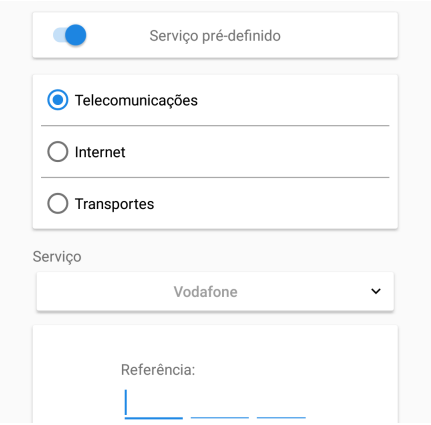


Figura 5.4: Pagamento de serviços pré-definido

Pagamento ao Estado

Esta funcionalidade tem como finalidade efetuar um pagamento de um imposto, multa, entre outros, utilizando como dados a respetiva referência e montante.

O ecrã é composto por dois botões no topo, destinados à facilitação do preenchimento dos dados. O primeiro encaminha o utilizador para a funcionalidade de pagamentos frequentes e o segundo encaminha o utilizador para a funcionalidade de deteção de uma fatura. Após os botões anteriores, é apresentado um fragmento destinado à seleção da conta de onde será feito o débito do montante do pagamento. Todas

estas funcionalidades anteriores serão explicadas mais a frente. De seguida, é apresentado um *CardView* destinado ao preenchimento de todas as informações relativas à fatura de pagamento. Este contém cinco *EditText* para o preenchimento da referência, onde se pretende que sejam colocados 3 dígitos em cada um e, dois *EditText* para o preenchimento do montante, sendo um correspondente aos cêntimos e outro aos euros. Por fim, apresenta-se um fragmento para selecionar a data de pagamento, aparecendo, por defeito, o dia em que este está a ser realizado e um botão para continuar o pagamento. Este ecrã é apresentado na Figura C.2a, em anexo.

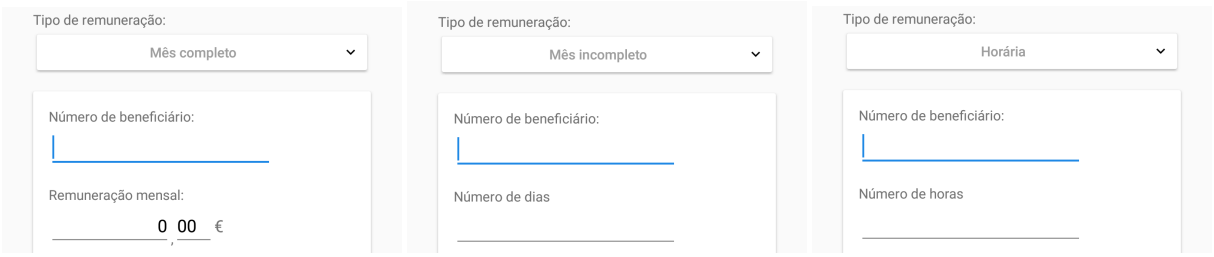
Quando o utilizador clica no botão "CONFIRMAR" é verificado, do lado do cliente, se os campos estão bem preenchidos. Ou seja, se o campo "Referência" possui 15 dígitos (3 em cada *EditText*) e se o montante é superior a 0,00 euros. Caso a validação anterior seja efetuada com sucesso, a aplicação faz um pedido ao servidor para validar o conteúdo do pagamento, enviando a entidade do estado (10095), a referência e o montante. O servidor por sua vez verifica se existe alguma fatura com a entidade e referência indicada, verificando também se o saldo da conta selecionada é suficiente para efetuar o pagamento. Caso a validação seja bem sucedida é apresentado, ao utilizador, um *AlertDialog* com as informações do pagamento, para que este as confirme. Caso a validação seja mal sucedida é apresentada uma mensagem de erro ao utilizador. Após o utilizador confirmar o pagamento e autenticar o mesmo através da matriz de autenticação ou da impressão digital, a aplicação envia para o servidor a informação do pagamento. Este recorre ao serviço de pagamentos, para persistir o pagamento na base de dados, subtrair ao saldo do utilizador o montante do pagamento e somar o mesmo ao saldo da entidade. Por fim, o servidor responde à aplicação com o id do pagamento, de modo a que seja possível emitir um comprovativo do mesmo. Este processo de obtenção de um comprovativo será explicado mais à frente.

Pagamento Segurança Social

Esta funcionalidade tem como finalidade efetuar um pagamento da segurança social. Este pagamento é mais complexo que os anteriores, pois requer mais dados.

O ecrã desta funcionalidade começa com um fragmento destinado à seleção da conta. A este fragmento seguem-se dois botões, um onde se seleciona o tipo de pagamento e outro para selecionar o tipo de remuneração. De seguida, apresenta-se um fragmento destinado ao preenchimento dos dados do pagamento. Este fragmento é constituída por dois *EditText*. O primeiro é destinado ao preenchimento do número de beneficiário da segurança social (NISS) e, o segundo, é destinado ao preenchimento da remuneração mensal, caso o tipo de remuneração selecionado seja "Mês completo", ao preenchimento dos dias de trabalhado, caso o tipo de remuneração selecionada seja "Mês incompleto" ou ao preenchimento do número de horas de trabalho, caso o tipo de remuneração seja "Horária". Estas várias opções para o preenchimento do segundo *EditText* podem ser vistas na Figura 5.5.

De seguida, no *CardView* é apresentado um *TextView*, onde é indicado o valor final do montante, calculado automaticamente através de um algoritmo baseado nas informações da segurança social [29]. Este montante só é calculado após o utilizador selecionar o ano e o mês nos quais pretende pagar a segurança social. Esta seleção é realizada a partir dos botões que se encontram abaixo do *CardView* anterior. Por último, no ecrã existe ainda um fragmento destinado à seleção da data de pagamento e um botão para dar continuação ao pagamento. Este ecrã é apresentado nas Figuras C.2b e C.2c, em anexo.



(a) Tipo de remuneração: Mês completo (b) Tipo de remuneração: Mês incompleto (c) Tipo de remuneração: Horária

Figura 5.5: Diferenças na seleção dos vários tipos de remuneração

Quando o utilizador clica no botão "CONTINUAR" é verificado, do lado do cliente, se os dois *EditText* do *CardView* foram preenchidos corretamente. Isto é, se o NISS é válido e se o campo seguinte não está vazio. Caso a verificação não encontre erros, a aplicação mostra ao utilizador um *AlertDialog* com as informações do pagamento, para que este confirme as mesmas. Caso a verificação encontre erros, é apresentado ao utilizador uma mensagem de erro que será explicada na Secção "Erros", mais à frente.

Quando o utilizador confirma o pagamento e se autentica com a matriz de autenticação ou com a impressão digital, a aplicação envia toda a informação do pagamento para o servidor, incluindo a entidade da segurança social (21056). O servidor, por sua vez, trata a informação, recorrendo ao serviço de pagamentos para persistir o pagamento na base de dados, subtrair ao saldo do utilizador o montante do pagamento e somar o mesmo ao saldo da entidade. Este persiste ainda as informações adicionais, características do pagamento da segurança social numa tabela dedicada, detalhada na Secção 4.3.6. Por fim, o servidor envia o id do pagamento ao utilizador para que seja possível emitir um comprovativo do pagamento. Este processo de obtenção de um comprovativo será explicado mais à frente. Se ocorrer um erro durante o procedimento anterior é exposta uma mensagem de erro e o pagamento não é realizado.

Fragmentos

Para evitar a repetição do código, como todos os tipos de pagamento utilizam os botões de selecionar a conta e selecionar a data de pagamento, foram implementados dois fragmentos. Estes serão incluídos dentro dos *layouts* dos diferentes tipos de pagamento, sendo a lógica de cada um destes tratada através de um *Presenter*. Isto faz com que o *Presenter* de cada tipo de pagamento já não tenha de tratar das ações do utilizador quando este seleciona uma conta ou uma data de pagamento.

De seguida é explicado o processo de implementação de cada um destes fragmentos.

Selecionar a conta

Este fragmento é composto por dois *TextView* e um botão, como se pode ver na Figura 5.6. O primeiro *TextView* mostra ao utilizador o saldo da conta que se encontra selecionada, o segundo dá *feedback* ao utilizador para que este perceba que o botão seguinte serve para selecionar uma conta diferente e o botão, como já foi mencionado, serve para o utilizador ver as várias contas que tem e selecionar a que pretende.

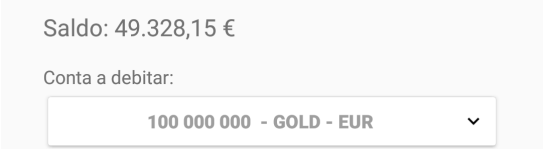


Figura 5.6: Fragmento para selecionar conta

Quando o utilizador carrega no botão para selecionar uma conta, é apresentado um *AlertDialog*, como pode ser visto na Figura 5.7. Este contém uma lista de cartões, representada através de uma vista *CoverFlow*. Esta corresponde a uma biblioteca externa que pode ser encontrada no *github* [24]. Nesta vista são apresentados os cartões do banco que, neste caso, são apenas protótipos desenhados com a ferramenta Pixlr. Para mudar de conta basta deslizar o dedo para o lado, fazendo com que os cartões deslizem para o lado também. O cartão da conta que será selecionado corresponde ao cartão em destaque, do qual se apresenta também a informação de saldo. Quando o utilizador carrega "OK", o id da conta selecionada será guardado nas *SharedPreferences*, de forma a que quando o utilizador navega na aplicação, isto é, muda de página, a conta que foi selecionada anteriormente continua selecionada na nova página. O saldo na *TextView* é atualizado para o saldo da conta selecionada no *AlertDialog*.

Quando um pagamento é efetuado, o id da conta selecionada neste fragmento é adquirido a partir das *SharedPreferences* e enviado para o servidor com as restantes informações do pagamento.

Figura 5.7: AlertDialog para selecionar conta

Selecionar a data

Este fragmento é composto apenas por uma *TextView* e um botão. Na TextView é dado, ao utilizador, o *feedback* acerca do botão abaixo e, o botão, serve para selecionar a data do pagamento. Este fragmento é apresentado na Figura seguinte (Figura 5.8).

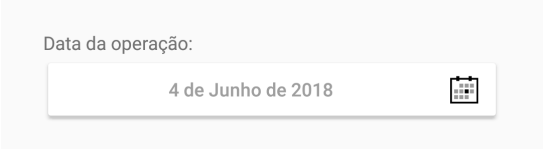


Figura 5.8: Fragmento para selecionar a data do pagamento

Quando o utilizador carrega no botão para selecionar a data do pagamento é-lhe apresentado um *DatePickerFragement* que corresponde basicamente a um calendário. Este é disponibilizado pela biblioteca do Android, tendo sido incluída uma restrição para que não possa ser selecionada uma data anterior ao dia presente. O *DatePickerFragement* é apresentado na Figura 5.9.

Quando o utilizador abre uma página de pagamento, a data de pagamento apresentada é, por defeito, o dia presente. Caso o utilizador selecione uma data distinta da apresentada, procede-se ao

armazenamento da mesma nas *SharedPreferences* e, posteriormente, quando o utilizador efetua um pagamento, a data armazenada anteriormente é recolhida.

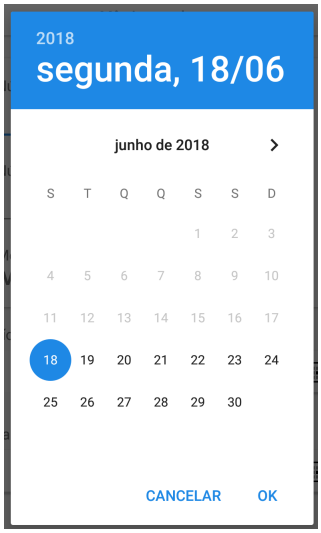


Figura 5.9: DatePickerFragment

Comprovativo do pagamento

Após um pagamento ser efetuado com sucesso, o utilizador é direcionado para uma página de onde é possível obter o comprovativo do pagamento. Esta página é apresentada através de um ecrã constituído por uma *TextView* e dois botões. Na *TextView* são apresentados as informações do pagamento realizado e, nos botões, as duas opções disponíveis de obter o respetivo comprovativo. A partir do primeiro botão é possível fazer diretamente o download do comprovativo e, no segundo botão, é possível fazer o envio por email do respetivo comprovativo. Este ecrã é apresentado na Figura C.3a.

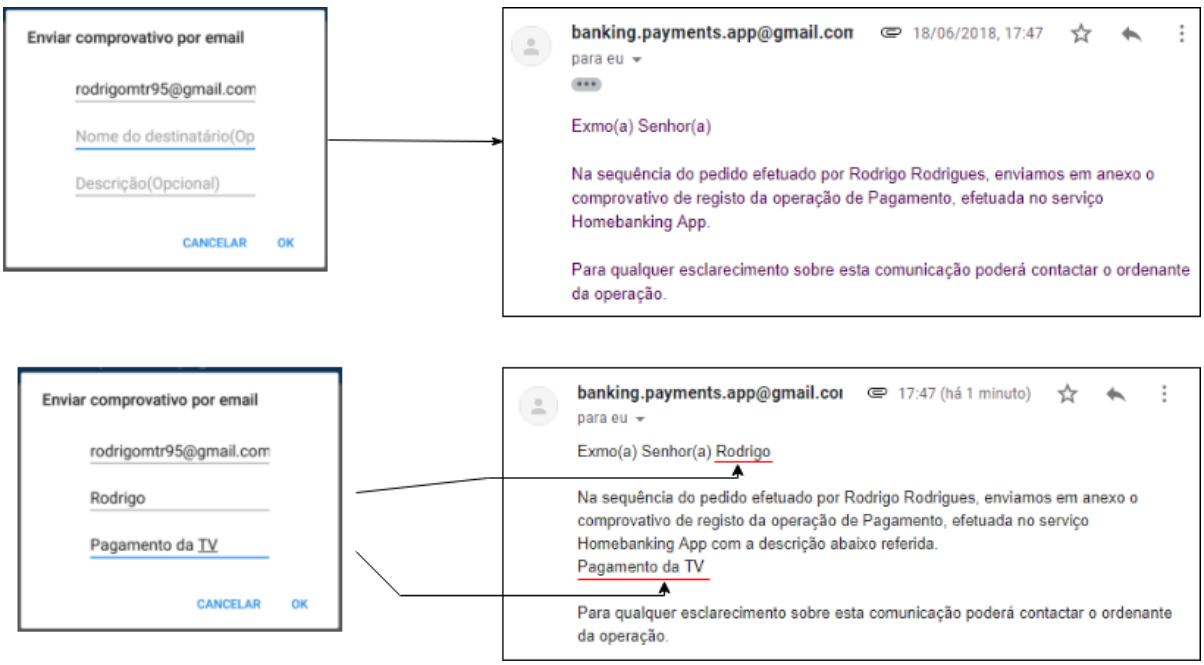
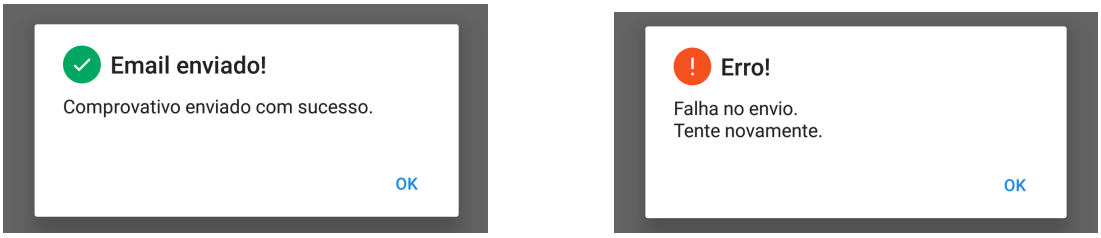


Figura 5.10: Alert Dialog para o envio do comprovativo por email (à esquerda) e diferença no preenchimento dos campos opcionais

Quando o utilizador carrega no botão de envio do comprovativo por email, é apresentado um *AlertDialog* com três *EditText*, para o preenchimento das informações necessárias para o envio do mesmo. No primeiro deve ser escrito o email de envio, sendo que este é pré-preenchido com o email do utilizador e pode ser alterado, caso o utilizador pretenda enviar o comprovativo para outro email. No segundo deve ser preenchido o nome do destinatário do email e, no terceiro, deve ser feita a descrição do pagamento. Os dois últimos *EditText* são campos opcionais. Na Figura 5.10 (figura anterior) podem ser verificadas as diferenças do email, quando os campos opcionais são preenchidos e quando não são. O comprovativo do pagamento corresponde a um ficheiro PDF e é apresentado na Figura D.4, em anexo. Quando o email é enviado com sucesso, é apresentado o *AlertDialog* presente na Figura 5.11a. Caso contrário, é apresentado o *AlertDialog* presente na Figura 5.11b.



(a) Sucesso no envio do comprovativo

(b) Erro no envio do comprovativo

Figura 5.11: AlertDialog com o resultado do envio do comprovativo por email

Quando o utilizador carrega num dos botões mencionados anteriormente é enviado um pedido ao servidor, contendo o id do pagamento, para se transferir ou enviar por email o comprovativo. Por sua vez, o servidor verifica se esse ficheiro existe, procurando-o através do id do pagamento (todos os ficheiros de comprovativo são guardados no servidor tendo como nome o id do pagamento). Caso o mesmo não exista, o servidor cria o ficheiro e transfere-o para o utilizador ou envia-o por email. A cada 10 minutos o servidor apaga todos os ficheiros de comprovativos que foram criados, evitando assim uma sobrecarga no espaço de armazenamento. Por fim, para voltar para a página principal, o utilizador pode carregar na seta presente na *toolbar* ou na seta de andar para trás do próprio dispositivo. Quando esta ação é efetuada, é apresentado ao utilizador um *AlertDialog* para que o mesmo confirme que quer sair da página (Figura 5.12). Este *AlertDialog* foi incluído pois, uma vez saindo desta página já não é possível obter um comprovativo do pagamento realizado.

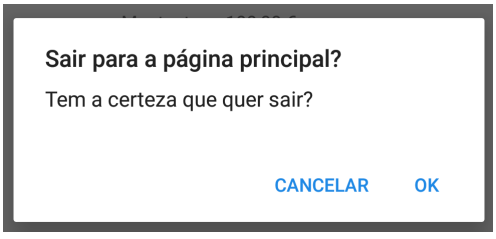


Figura 5.12: Alert Dialog para voltar para a página principal

Pagamento por *QR Code*

Esta funcionalidade tem como objetivo facilitar o preenchimento dos campos dos dados do pagamento, preenchendo-os automaticamente. Assim, os campos "Entidade", "Referência" e "Montante" são preenchidos automaticamente, recorrendo ao reconhecimento de um *QR Code*, presente numa fatura de

pagamento. Esta funcionalidade não pode ser utilizada na prática, pois as faturas ainda não possuem esse tipo de tecnologia.

O ecrã desta funcionalidade é composto apenas por uma *SurfaceView*, que mostra a câmara e uma imagem com uma mira, para que o utilizador perceba onde deve posicionar o dispositivo de forma a que este detete o *QR Code* (Figura 5.13). Para a deteção do *QR Code* foi utilizada a biblioteca *Mobile Vision* da Google [17]. Esta disponibiliza uma API com funções que detetam o conteúdo de um *QR Code*. O conteúdo é convertido para uma String que depois é validada. O *QR Code* de uma fatura segue o formato "Entidade":"Referência":"Montante". Por exemplo, o *QR Code* presente na fatura fictícia da MEO (Anexo E), contém o texto "21159:123456789:56,40". Para a validação deste, a String resultante da deteção é separada por ":", utilizando a função *split*. Esta cria um vetor de strings [13], sendo que a posição 0 é associada a uma variável chamada "entity", a posição 1 é associada a uma variável chamada "ref" e a posição 2 é associada a uma variável chamada "amount". De seguida, verifica-se a validade de cada variável da forma seguinte:

- A variável "entity" corresponde à *regex* "^[\\d]{5}$" (exatamente 5 dígitos);
- A variável "ref" corresponde à *regex* "^[\\d]{9}$" (exatamente 9 dígitos);
  - Ou se a variável "entity" for igual a "10095" (entidade do estado) corresponde à *regex* "^[\\d]{15}$" (exatamente 15 dígitos);
- A variável "amount" corresponde à *regex* "[\\d]+[.|,][\\d]+" (um ou mais dígitos seguidos por um ponto ou vírgula seguido por um ou mais dígitos).

Quando todas as variáveis são validadas com sucesso, estas são transferidas para a *Activity* do "Pagamento de Serviço" ou "Pagamento ao Estado". O utilizador é direcionado para uma destas páginas com os campos "Entidade", "Referência" e "Montante" já preenchidos (ou apenas os últimos dois no caso de um pagamento ao estado). Quando o *QR Code* não é válido é apresentada uma mensagem de erro ao utilizador e este é redirecionado para o menu de pagamentos, novamente.

Figura 5.13: Mira presente no ecrã de reconhecimento por *QR Code*

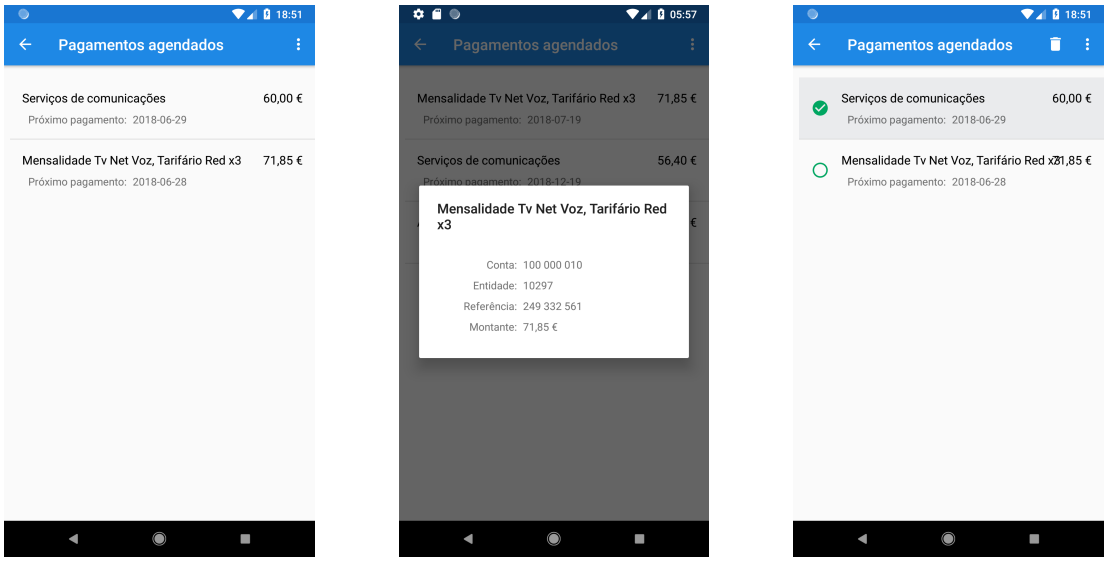
Pagamentos Agendados

Esta funcionalidade tem como objetivo fazer pagamentos de forma regular automaticamente. Quando um utilizador faz o pagamento de um serviço e seleciona um tipo de periodicidade, esse pagamento é realizado automaticamente em intervalos de tempo iguais à periodicidade antes selecionada. A periodicidade de um pagamento pode ser "Semanal", "Quinzenal", "Mensal", "Trimestral", "Semestral" ou "Anual".

Para que isto aconteça, quando o pagamento é realizado e selecionada uma determinada periodicidade, o servidor persiste na tabela *Periodic_Payment* (Secção 4.3.5) o id do pagamento, a periodicidade do mesmo, a data do próximo pagamento e o id do utilizador que efetuou o pagamento. Para que os pagamentos sejam realizados periodicamente, o servidor a cada 24 horas verifica na tabela mencionada

anteriormente, se existe algum pagamento para ser efetuado no dia corrente. Quando existe um pagamento para ser efetuado, é verificado, em primeiro lugar, se o utilizador tem saldo suficiente para a realização do mesmo. Caso este tenha saldo suficiente, o pagamento é executado, sendo feita a transação do valor do montante, para a respetiva entidade. Após a realização do pagamento é enviado um email para o utilizador com o comprovativo do pagamento (Figura D.2, em Anexo) e uma notificação para o dispositivo do utilizador com a informação de que o pagamento foi realizado, esta é detalhada mais à frente na Secção 5.1.2. Por fim, a data do próximo pagamento é atualizada na base de dados de acordo com a periodicidade. Por exemplo, se o pagamento foi feito dia 19-06-2018 e a periodicidade é "Mensal", então a data do próximo pagamento é atualizada para 19-07-2018. Caso o utilizador não possua saldo suficiente na conta para efetuar este pagamento, este não é efetuado e é enviado um email ao utilizador a informá-lo acerca do sucedido. Este email pode ser visto no Anexo D.3.

Na aplicação é possível verificar os pagamentos que foram agendados. Estes podem ser vistos através do menu de pagamentos, na opção "Pagamentos agendados" (Figura C.1d). O ecrã dos "Pagamentos agendados" contém uma lista com pagamentos que foram agendados pelo utilizador, esta é apresentada na Figura 5.14a, com dois pagamentos de exemplo. Quando se clica num item desta lista, é apresentado um *AlertDialog* com a informação do pagamento (Figura 5.14b). O utilizador pode também selecionar e eliminar os mesmos. Para selecionar é necessário pressionar o item até que este mude de cor e apareça, do lado esquerdo, um *checkbox*. Para selecionar mais pagamentos basta clicar, uma vez, nos pagamentos pretendidos. Para eliminar um ou mais pagamentos, estes devem estar selecionados e basta clicar no ícone com o caixote do lixo, presente na *toolbar* (Figura 5.14c). Quando um pagamento é eliminado, a sua realização automática deixa de ser efetuada.



(a) Ecrã "Pagamentos Agendados" (b) Informação do pagamento (c) Seleção de um pagamento

Figura 5.14: Pagamentos agendados

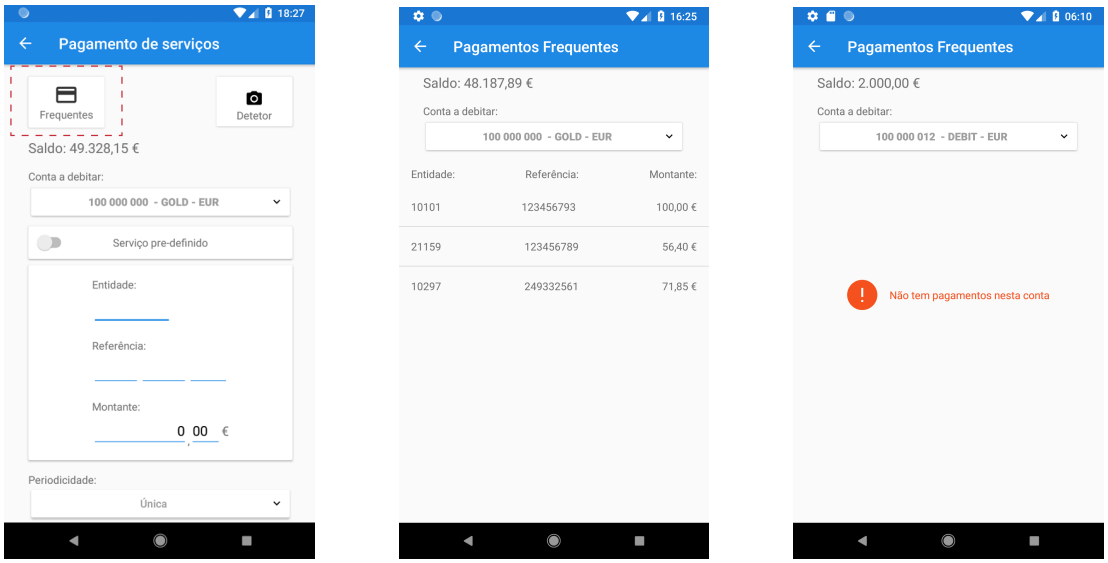
Pagamentos frequentes

Esta funcionalidade tem como objetivo facilitar um pagamento que é efetuado frequentemente, apresentando ao utilizador uma lista de pagamentos que este efetua frequentemente. O utilizador pode selecionar um destes pagamentos de maneira a que os campos "Entidade", "Referência" e "Montante" sejam pré-

preenchidos na página de pagamento.

A página dos pagamentos frequentes é acedida a partir da página "Pagamento de Serviços" ou "Pagamentos ao Estado", carregando no botão "Frequentes" (Figura 5.15a). Quando o utilizador entra nesta página (Figura 5.15b) é feito um pedido ao servidor para que este envie os pagamentos frequentes da conta que está selecionada. O servidor, por sua vez, ordena os pagamentos numa lista por ordem decrescente dos pagamentos com maior número de ocorrências e, envia-a para a aplicação.

No ecrã dos pagamentos frequentes, os itens são dispostos em lista, apresentando a respetiva entidade, referência e montante do pagamento. Acima desta lista, encontra-se um fragmento destinado à seleção da conta. Quando o utilizador seleciona uma conta distinta, todo o processo anterior é repetido. Quando a conta selecionada não tem pagamentos frequentes, é apresentada a mensagem de erro presente na Figura 5.15c.



(a) Botão "Frequentes" a tracejado (b) Ecrã inicial (c) Mensagem de erro

Figura 5.15: Pagamentos frequentes

Reconhecimento de uma fatura

O reconhecimento de uma fatura consiste na deteção dos dígitos da entidade, referência e montante presentes no campo "Pagamento por multibanco" da fatura, preenchendo-os automaticamente, na página do pagamento, com os dígitos detetados.

Esta funcionalidade já está disponível em algumas aplicações de bancos utilizados em Portugal, como é o caso do Santander Totta e do Novo Banco, sendo que nesta a deteção é realizada de forma ligeiramente diferente. Nestas é necessário tirar uma fotografia à fatura e futuramente é realizada a deteção dos dados a partir da fotografia. Este método faz com que seja necessário ocupar memória do dispositivo, de modo a se realizar o reconhecimento da fatura desejado.

No caso da aplicação *homebanking* isso não acontece, pois basta apenas apontar com a câmara para a fatura, para que a deteção seja realizada. Esta funcionalidade funciona tanto para o "Pagamento de Serviços", como para o "Pagamento ao Estado", e pode ser acedida através da página de um destes pagamentos, carregando no botão "Detetor". Na Figura 5.16 é apresentado o botão "Detetor" da página dos "Pagamento de Serviços".

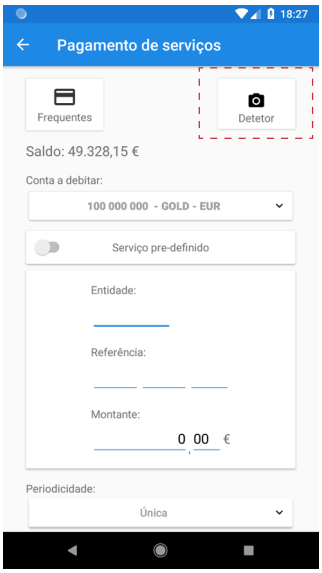
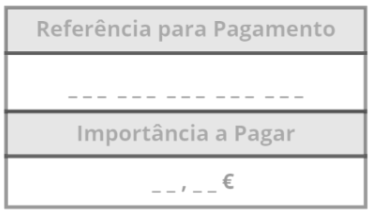


Figura 5.16: Botão "Detetor" a tracejado

Quando o utilizador carrega no botão anterior é lhe apresentado um ecrã constituído por uma *SurfaceView*, que mostra a câmara e, por uma de duas imagens. A primeira, Figura 5.17a, é apresentada quando o utilizador se encontra na página "Pagamento de Serviços" e, a segunda, Figura 5.17b, é apresentada quando o utilizador acede a esta funcionalidade a partir da página "Pagamento ao Estado". Estas servem para ajudar o utilizador a posicionar a câmara do dispositivo na área correta, para que o "Detetor" seja capaz de detetar as informações da fatura.

(a) Pagamento de Serviços



(b) Pagamentos ao Estado

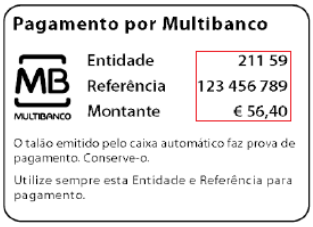
Figura 5.17: Imagens de ajuda no processo de deteção dos dados de uma fatura

Para o reconhecimento da fatura foi utilizada a biblioteca *Mobile Vision* da Google [18], que disponibiliza uma API com funções que detetam texto a partir da câmara do dispositivo. O conteúdo do reconhecimento é convertido para uma String, que depois é validada.

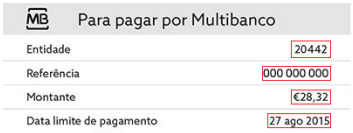
Antes da implementação do algoritmo para a validação e reconhecimento da fatura, foram feitos vários testes para perceber como era detetado o conteúdo da fatura quando se apontava a câmara para o campo "Pagamento por Multibanco".

Dos testes realizados obtiveram-se as seguintes conclusões:

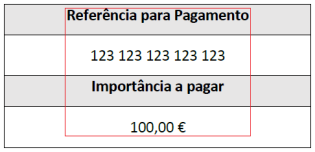
- Na leitura das faturas de serviços, o texto pode ser detetado de duas maneiras: em bloco (Figura 5.18a) ou em linha (Figura 5.18b), dependendo da fatura;
- Na leitura das faturas do estado, o texto é sempre detetado em bloco (Figura 5.18c).



(a) Deteção, em bloco, de uma fatura de serviços



(b) Deteção, em linha, de uma fatura de serviços



(c) Deteção, em bloco, de uma fatura do estado

Figura 5.18: Tipos de deteção

Após a realização dos testes mencionados anteriormente, o algoritmo de validação que foi implementado tem em conta as conclusões retiradas. Assim, o processo de validação de uma fatura de um serviço é feito da seguinte forma:

1. A String com a deteção é separada por mudanças de linha "\n" e é criado um vetor em que cada posição corresponde a uma linha.

2. Se o tamanho do vetor for maior ou igual a 3 (deteção do bloco):
   (a) A posição 0 corresponde à entidade, se a string corresponder ao *regex* "^[\\d]{5}$" (exatamente 5 dígitos);
   (b) A posição 1 corresponde à referência, se a string corresponder ao *regex* "^([\\d]{3}[ ]?){2}\\d{3}$" (exatamente 9 dígitos, podendo estar separados por espaços a cada 3 dígitos);
   (c) A posição 2 corresponde ao montante, se a string corresponder ao *regex* "[\\d]+[.|,][\\d]+ ?€" (um ou mais dígitos seguidos por um ponto ou vírgula, seguido por um ou mais dígitos e o símbolo da moeda euro no final);

3. Se o tamanho do vetor for menor a 3 (deteção da linha):
   (a) A deteção corresponde à entidade, se a string corresponder ao *regex* "^[\\d]{5}$" (exatamente 5 dígitos);
   (b) A deteção corresponde à referência, se a string corresponder ao *regex* "^([\\d]{3}[ ]?){2}\\d{3}$" (exatamente 9 dígitos, podendo estar separados por espaços a cada 3 dígitos);
   (c) A deteção corresponde ao montante, se a string corresponder ao *regex* "[\\d]+[.|,][\\d]+ ?€" (um ou mais dígitos seguidos por um ponto ou vírgula, seguido por um ou mais dígitos e o símbolo da moeda euro no final);

Em relação às faturas do estado, estas são validadas da seguinte forma:

1. A String com a deteção é separada por mudanças de linha "\n" e é criado um vetor em que cada posição corresponde a uma linha.

2. Se o tamanho do vetor for igual a 4 (deteção do bloco):
   (a) A posição 1 corresponde à referência, se a string corresponder ao *regex* "^([\\d]{3}[ ]?){4}\\d{3}$" (exatamente 15 dígitos, podendo estar separados por espaços a cada 3 dígitos);
   (b) A posição 3 corresponde ao montante, se a string corresponder ao *regex* "[\\d]+[.|,][\\d]+ ?€" (um ou mais dígitos seguidos por um ponto ou vírgula, seguido por um ou mais dígitos e o símbolo da moeda euro no final);

(c) As posições 0 e 2 correspondem ao texto “Referência para Pagamento” e “Importância a pagar”, respetivamente, e por isso são ignoradas.

Após esta validação, é verificado se as variáveis entidade, referência e montante (apenas as últimas duas no caso de um pagamento ao estado) não são nulos. Caso isto se confirme o utilizador é direcionado para a página de pagamento com os campos já preenchidos. Caso contrário a deteção continua.

Notificações

Quando um pagamento, que foi agendado pelo utilizador, ocorre, este recebe uma notificação a informar que o pagamento foi efetuado. Esta notificação é apresentada nas Figuras 5.19 e 5.20.

Figura 5.19: Ícone da notificação na barra de status

Figura 5.20: Notificação na central de notificações

Para a implementação deste processo é utilizada a API *Firebase Cloud Messaging* [20], que fornece dois serviços úteis para o envio de notificações. O primeiro é o registo de um *token* que tem, como finalidade, identificar um dispositivo. Este segue os seguintes passos, que são representados, também, através da Figura 5.21:

1. O dispositivo envia para o FCM o *Sender ID* (id do projeto criado no firebase), a *API Key* (chave que autentica o servidor do firebase permitindo assim acesso ao mesmo) e o *App ID* (id da aplicação).

2. Quando o utilizador instala a aplicação pela primeira vez, o FCM envia para o dispositivo um *token* de registo, que tem como finalidade identificar o dispositivo de maneira a que este possa começar a receber notificações a partir do FCM. A aplicação recebe este *token*, a partir de uma classe que implementa a classe *FirebaseInstanceIdService* e, sempre que o mesmo é atualizado, esta chama uma função que guarda o *token* nas *SharedPreferences*.

3. Quando o utilizador entra na aplicação pela primeira vez e se autentica, esta irá obter o *token*, a partir das *SharedPreferences*, e enviá-lo para o servidor.

4. Por sua vez, o servidor persiste na base de dados o *token* de registo.

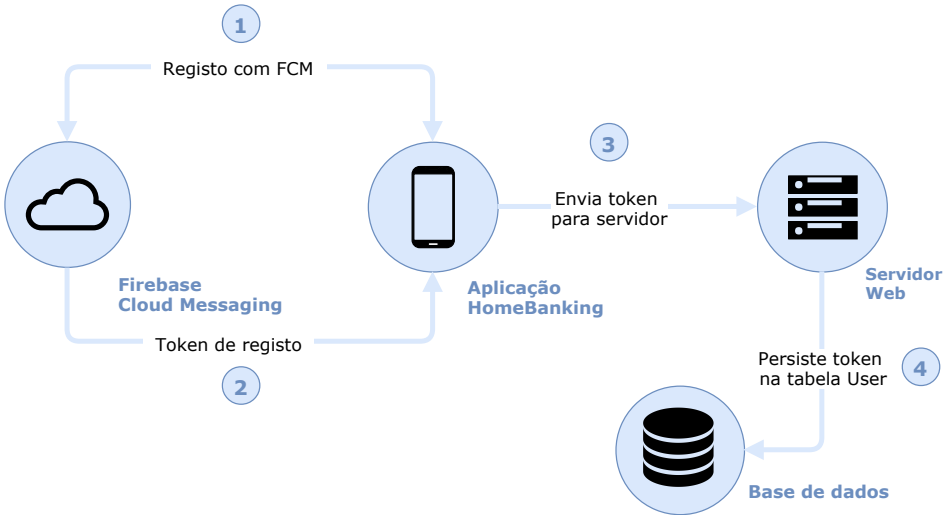


Figura 5.21: Esquema do envio de notificações baseado em [19]

O segundo serviço consiste no envio de notificações, de modo a que o utilizador receba notificações da aplicação, mesmo quando esta não está a ser utilizada. Este segue os seguintes passos, que são representados, também, através da Figura 5.22:

1. No dia de realização de um pagamento agendado, para além dos passos descritos em "Pagamentos Agendados", o servidor faz uma *query* à base de dados de forma a obter o *token* do utilizador que agendou o pagamento.

2. De seguida, o servidor cria um objeto JSON contendo o *token* do destinatário, o título, o corpo da notificação e os dados do pagamento.

3. O FCM trata de enviar a notificação para o dispositivo do utilizador, que a recebe a partir do serviço *FirebaseMessagingService*. Este serviço é implementado por uma classe da aplicação e, sempre que o FCM envia uma notificação, esta chama uma função que recebe os dados da notificação.

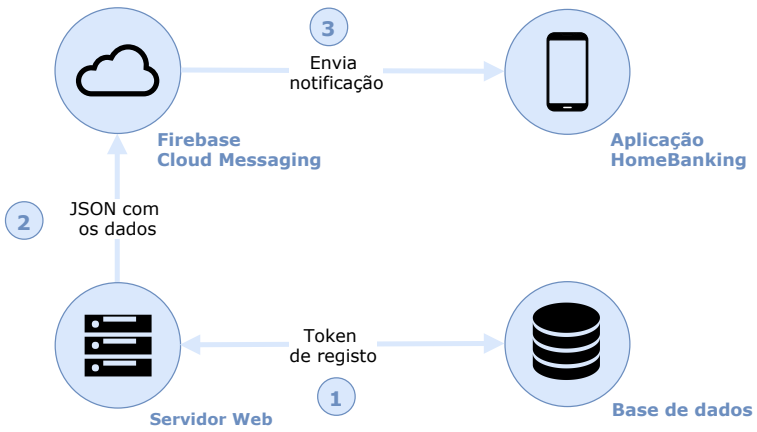


Figura 5.22: Esquema do envio de notificações baseado em [19]

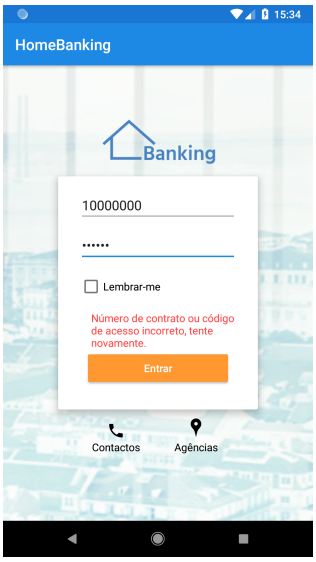
Posto isto, o utilizador recebe uma notificação e quando clica na mesma é redirecionado para a página de login da aplicação. Após efetuar o login é-lhe apresentada a página de comprovativo de pagamento.

Erros

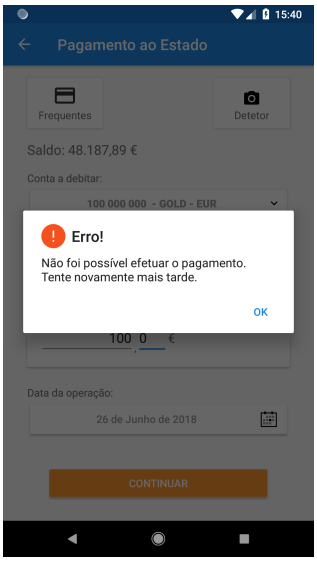
Na aplicação, os erros são apresentados de três formas distintas:

- Se um campo for mal preenchido, a linha do *EditText* fica vermelha e abaixo desta é apresentada a descrição do erro. Este erro é apresentado na Figura 5.23a;
- Se houver um erro do lado do servidor, na validação de um pagamento ou na validação do login, o utilizador recebe uma descrição do erro acima do último botão clicado. Este erro é apresentado na Figura 5.23b;
- Outros erros do lado do servidor, em certos casos, são representados através de um *AlertDialog*. Um exemplo deste erro é apresentado na Figura 5.23c.

(a) Campo mal preenchido



(b) Erro de validação



(c) AlertDialog de Erro

Figura 5.23: Tipos de erros

Bilingue

A aplicação suporta as línguas inglesa e portuguesa. Quando o utilizador define português como idioma do seu dispositivo, a aplicação é apresentada em português, caso contrário, é apresentada em inglês.

Para estas duas opções de idioma foi necessário criar dois ficheiros XML na pasta "strings" (Figura 5.25 à esquerda). Estes ficheiros contêm conjuntos chave-valor, em que a chave é o nome que identifica a String e o valor é o conteúdo da mesma. Um exemplo de um item destes ficheiros podem ser vistos nas Figuras 5.24a e 5.24b. Estes itens são invocados a partir dos seus nomes, sendo que quando o idioma definido no dispositivo do utilizador é o português, é utilizado o ficheiro da pasta values-pt-rPT e, em qualquer outro caso é utilizado o ficheiro da pasta values (Figura 5.25 à direita).

```
<string name="amount">Amount:</string>
```

(a) strings.xml

```
<string name="amount">Montante:</string>
```

(b) strings.xml(pt-rPT)

Figura 5.24: Item dos ficheiros "strings.xml"

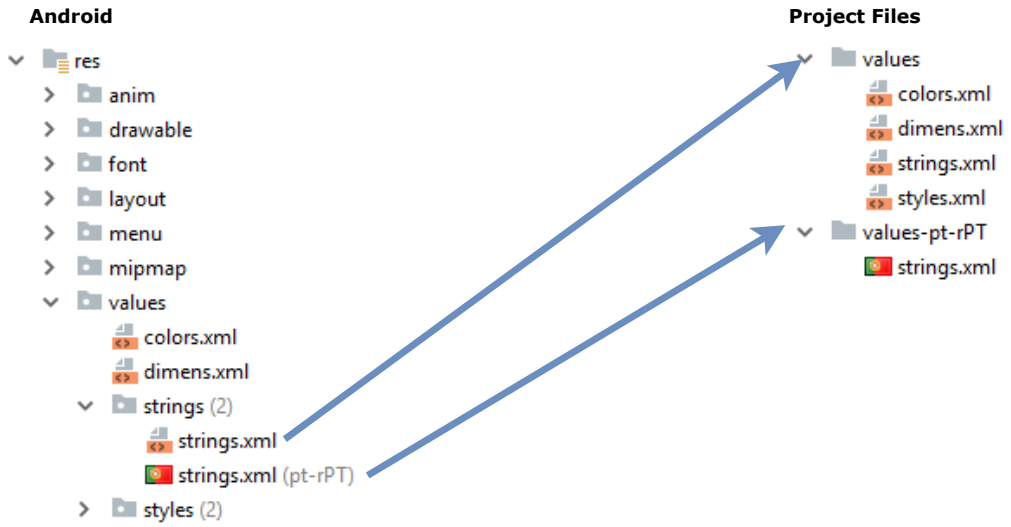


Figura 5.25: Ficheiros para suporte bilingue

5.1.3 Migração para Kotlin

A Google anunciou na conferência Google I/O 2017 que a linguagem de programação Kotlin iria passar a ser uma linguagem oficial para o desenvolvimento de aplicações para Android [7]. Desta forma e, como a aplicação já estava toda implementada em linguagem Java, decidiu-se fazer a migração de todo o código Java para Kotlin. A linguagem Kotlin foi desenvolvida pela JetBrains (empresa que desenvolveu a IntelliJ IDEA, na qual o Android Studio se baseia) e a sua primeira versão foi lançada em 2016. Esta linguagem possui uma serie de pontos fortes:

- Concisa: É uma linguagem pouco textual. As data classes são um exemplo disto, apenas com a declaração desta são criados instantaneamente *getters*, *setters*, e as funções *equals()*, *hashCode()*, *toString()* e *copy()*. Na Figura 5.26 é possível verificar as diferenças entre uma classe de dados em Java e uma em Kotlin.

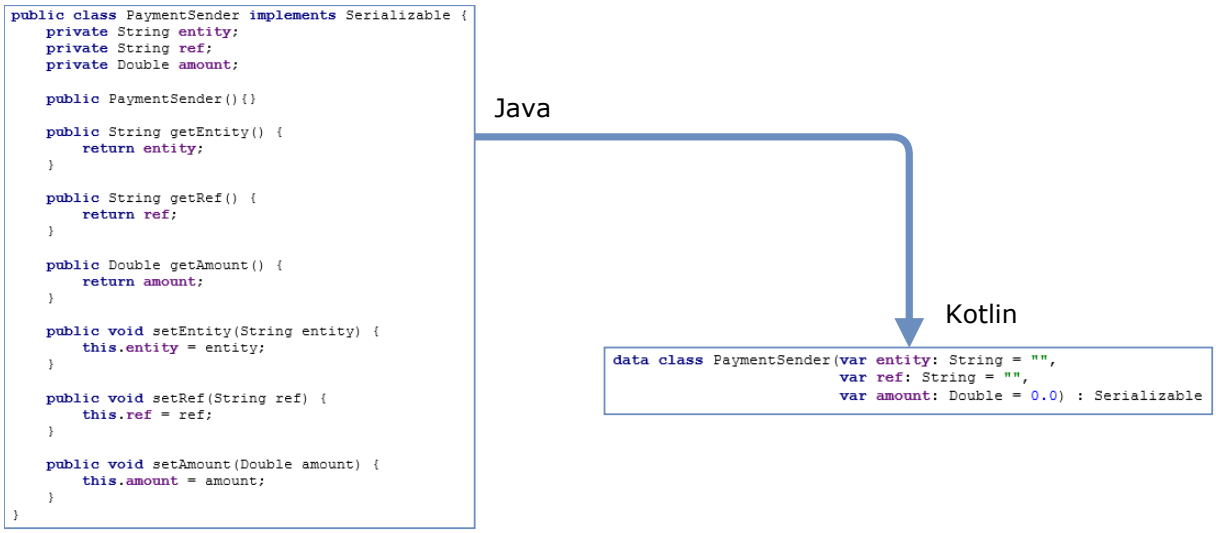


Figura 5.26: Diferença entre Java e Kotlin em objetos

- Segura: É *null safe*, ou seja, evita a possibilidade de ocorrerem erros de *null pointer exception* [22].

- Interoperável: Esta linguagem corre sobre a JVM e por isso é 100% compatível com as bibliotecas desta.

- *Tool-friendly*: Esta linguagem foi criada por uma empresa de ferramentas de edição de código (IDE) e, por esse motivo, a sua edição é muito facilitada por estas ferramentas.

Para a migração foi utilizada uma ferramenta do Android Studio que converte o código Java para Kotlin (Figura 5.27). Existem extensões do Kotlin para Android que facilitam a ligação das classes às componentes do *layout*, deixando de ser necessário chamar o método *"findViewById"* para fazer esta ligação. Assim, torna-se apenas necessário colocar o id do componente, como se pode verificar no exemplo da Figura 5.28. Quando o código é convertido são feitas algumas alterações de maneira a simplificar ainda mais o mesmo. Um exemplo disto pode ser visto na Figura 5.29.

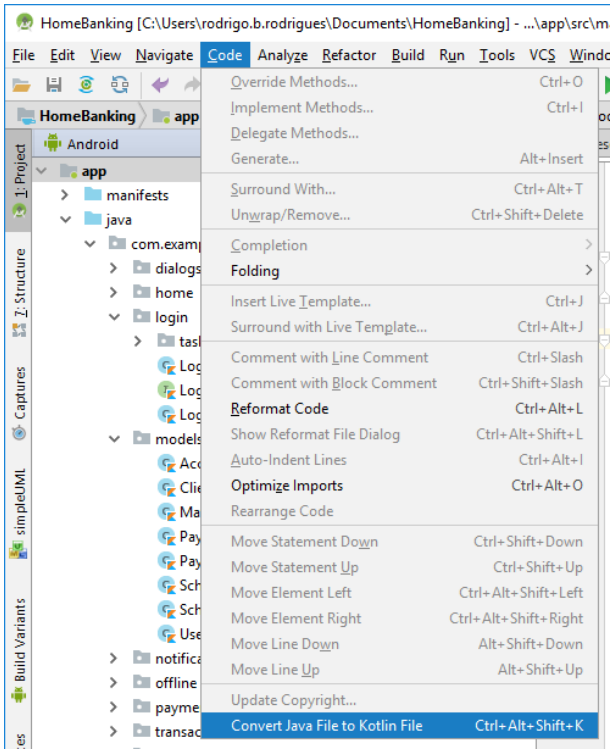


Figura 5.27: Ferramenta para converter o código Java para Kotlin

```java
entityET = findViewById(R.id.entity_entry);

refET1 = findViewById(R.id.ref_entry_1);
refET2 = findViewById(R.id.ref_entry_2);
refET3 = findViewById(R.id.ref_entry_3);

amountET1 = findViewById(R.id.amount_entry_1);
amountET2 = findViewById(R.id.amount_entry_2);

setETTextChange(entityET, refET1);
setETTextChange(refET1, refET2);
setETTextChange(refET2, refET3);
setETTextChange(refET3, amountET2);
setETTextChange(amountET2, amountET1);
```

Java

Kotlin

```kotlin
setETTextChange(entity_entry, ref_entry_1)
setETTextChange(ref_entry_1, ref_entry_2)
setETTextChange(ref_entry_2, ref_entry_3)
setETTextChange(ref_entry_3, amount_entry_2)
setETTextChange(amount_entry_2, amount_entry_1)
```

Figura 5.28: Ligação às componentes do layout

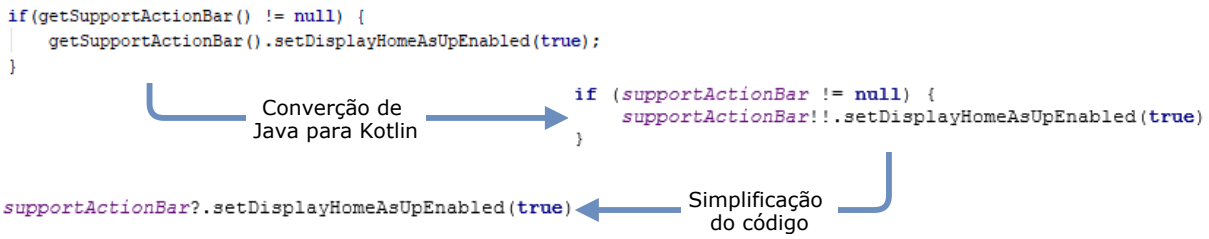


Figura 5.29: Simplificação do código após a conversão automática

Neste processo de migração, foram migradas 95 classes tendo sido despendidas 28 horas para tal. A diferença entre as versões Java e Kotlin é marcada pela densidade de texto e também pelo espaço de armazenamento necessário. Estas diferenças podem ser verificadas na Tabela 5.1. Desta tabela conclui-se que a linguagem Kotlin é menos textual que a linguagem Java.

Tabela 5.1: Algumas diferenças entre as versões Java e Kotlin

	Classes	Tamanho (kB)	Linhas	max lin	min lin	med lin
Java	95	321	9350	1218	9	97
Kotlin	95	213	6083	255	6	64
Diferença	0	108	3267	963	3	33

5.2 Servidor Web

O servidor foi desenvolvido na linguagem Java, recorrendo à *framework Spring*. Este consiste num *RESTful Webservice* que recebe e responde a pedidos efetuados pela aplicação móvel e, contém as seguintes componentes:

***Controller*:** Recebe os pedidos do cliente, recorre aos serviços para resolver o pedido e envia a resposta ao cliente.

***Service*:** É onde são resolvidos os pedidos, ou seja, efetua a lógica e quando necessário, comunica com a base de dados a partir da componente *Repository*. Por exemplo, executar um pagamento, subtraindo o valor do montante ao saldo do utilizador e somando o mesmo ao saldo da entidade, e de seguida persistir na base de dados os dados do pagamento.

***Rpository*:** Esta componente é responsável pela comunicação com a base de dados recorrendo à interface *JPARepository* [10] e às classes da componente *Entity*.

***Configuration*:** É responsável pelas configurações de segurança utilizadas para a autenticação do utilizador e para a encriptação das palavras-passe.

***Components*:** Responsável por realizar eventos temporários. Esta a cada 24 horas verifica se existem pagamentos agendados que precisam de ser realizado e, a cada 10 minutos, apaga os ficheiros dos comprovativos de pagamentos, evitando assim que haja sobrelotação da memória do servidor. Também é responsável pelo envio de emails recorrendo à interface *JavaMailSender*[9].

***Entity*:** Esta componente faz parte do JPA (Java Persistence API)[8] e contém classes que caracterizam as tabelas da base de dados, facilitando a forma como são feitas as *queries* à base de dados.

5.3 Base de dados

Foi utilizada a tecnologia *SQL Server* para o desenvolvimento da base de dados. Nesta foi utilizada a ferramenta SQL Server Management Studio 2017 para ajudar a desenvolver a base de dados, que foi implementada de acordo com o modelo da Secção 4.3.

Capítulo 6

Testes

Durante o decorrer da implementação foram realizados testes para avaliar o funcionamento das componentes que iam sendo desenvolvidas. Além destes pequenos testes, no final da implementação foram realizados testes de usabilidade, de reconhecimento de faturas e, também, de compatibilidade. Todos estes testes serão analisados em detalhe nas próximas secções.

6.1 Testes de Usabilidade

Os testes de usabilidade consistem no envolvimento de voluntários a utilizar a aplicação, sendo dadas várias tarefas para estes realizarem. Cada tarefa proposta pretende testar uma funcionalidade.

Nestes testes participaram 13 voluntários, com idades compreendidas entre os 22 e os 56 anos, sendo 46% do sexo feminino e 54% do sexo masculino. Foram definidas 13 tarefas para cada voluntário realizar, que são apresentadas na Tabela 6.1. Após a realização de cada uma destas foi pedido a cada voluntário para avaliar a dificuldade sentida na realização da mesma, tendo em conta a cotação da Tabela 6.2 e deixar alguns possíveis comentários de melhoria.

Tabela 6.1: Tarefas propostas a cada voluntário

Tarefa	Descrição
1	Efetuar login com os dados fornecidos
2	Efetuar o pagamento de uma dada fatura (Serviço), utilizando na autenticação a impressão digital
3	Efetuar o pagamento de uma dada fatura (Estado), utilizando na autenticação a matriz de autenticação
4	Efetuar o pagamento de uma dada fatura a partir do QR Code
5	Efetuar o pagamento da segurança social
6	Agendar o pagamento de um serviço para que este seja feito mensalmente
7	Apagar agendamento do pagamento efetuado anteriormente
8	Realizar um pagamento a partir de pagamentos frequentes
9	Realizar um pagamento de serviços a partir do reconhecimento de uma fatura
10	Realizar um pagamento ao estado a partir do reconhecimento de uma fatura
11	Enviar o comprovativo por email
12	Efetuar o download do comprovativo
13	Efetuar logout

Tabela 6.2: Cotação da dificuldade sentida na realização de uma tarefa

Dificuldade					
Muito fácil	Fácil	Médio	Difícil	Muito difícil	Não consegui
5	4	3	2	1	0

A realização de algumas tarefas necessitava de alguns dados que foram fornecidos aos voluntários. Para a realização da tarefa 1, foi dado ao utilizador um número de contrato e uma senha de acesso. Para a tarefa 3, foi dada a matriz presente no Anexo D.1. Por fim, para as tarefas 4, 9 e 10 foram fornecidas as faturas presentes no Anexo E.

A partir das cotações dadas pelos voluntários foi produzido o gráfico da Figura 6.1, onde foram introduzidas as cotações média, mínima e máxima de cada tarefa. A partir da observação do gráfico mencionado verifica-se que a funcionalidade presente na tarefa 1 não necessita de alterações, pois esta foi efetuada facilmente por todos os voluntários, tendo sido a tarefa com melhor *feedback*. As funcionalidades presentes nas tarefas 4, 8, 10, 11, 12 e 13 necessitam de poucas alterações, pois as cotações obtidas variam entre 4 e 5. As funcionalidades presentes nas tarefas 2, 3, 6, 7 e 9 precisam de um pouco mais de alterações, pois já obtiveram cotações entre 3 e 5. Por fim, a funcionalidade da tarefa 5 necessita de muitas alterações ou mesmo uma reformulação, pois corresponde a funcionalidade com pior cotação, tendo obtido a cotação mínima igual a 2.

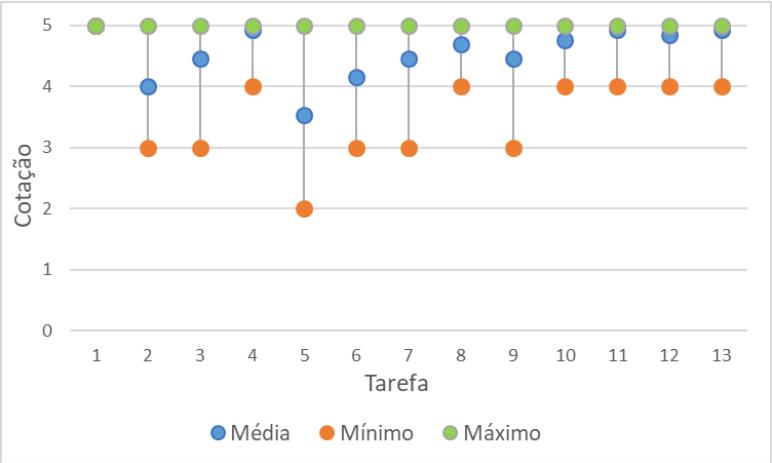


Figura 6.1: Gráfico das cotações médias, mínimas e máximas obtidas para cada tarefa

De seguida, serão apresentadas as observações feitas pelos voluntários, à medida que iam realizando as tarefas:

- Observação: Preenchimento do montante deveria ser feito como no multibanco, isto é, quando se insere um dígito o mesmo deveria deslizar para a esquerda.

  Tarefas afetas: 2, 3

  Explicação: Quando o utilizador finaliza o preenchimento do campo da referência é direcionado para o preenchimento do montante com o cursor colocado na parte à direita da vírgula. A maioria dos voluntários efetuou esta operação de forma errada devido à sua experiência no preenchimento do montante no multibanco. Seguindo a experiência indicada anteriormente, estes começavam pelo

preenchimento das unidades do montante presente na fatura no campo à direita da vírgula (Passo 2 da Figura 6.2), esperando que estes números deslizassem para a esquerda da vírgula (Passo 3 da Figura 6.2). Ao invés do deslizamento dos números, o cursor é movido para a esquerda, deixando o valor correspondente aos euros na zona correspondente aos cêntimos. Os utilizadores para a finalização do preenchimento do montante colocavam o valor dos cêntimos no local correspondente aos euros (Passo 4 da Figura 6.2).

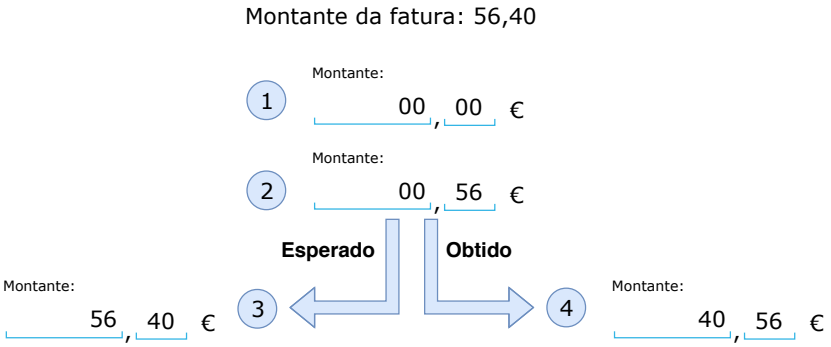


Figura 6.2: Resultado esperado

- Observação: As colunas da matriz de autenticação deveriam começar em 1.

  Tarefas afetas:2, 3, 4, 5, 6, 8, 9, 10

  Explicação: A Matriz de autenticação é enviada para o utilizador de acordo com o email que se pode ver no Anexo D.1, as colunas são identificadas com números de 0 a 4. Isto confundiu alguns voluntários, pois estes esperavam que as colunas começassem com o número 1 e, quando era pedida uma posição na coluna 1, por exemplo A1, estes inseriam o valor da posição A0.

- Observação: No pagamento da segurança social deveria ser possível selecionar o mês antes do ano.

  Tarefas afetas: 5

  Explicação: Quando os voluntários tinham de selecionar o período de pagamento (Figura C.2c), a maioria deles tentava primeiro carregar no botão "Mês", mas este botão encontra-se bloqueado de maneira a obrigar o utilizador a selecionar primeiro o ano. Isto foi feito desta forma, pois na pesquisa realizada às outras aplicações deste género, este processo é realizado desta forma.

- Observação: Nos "Pagamentos agendados" devia ter uma opção de criar o pagamento agendado.

  Tarefas afetas: 6

  Explicação: Durante a realização da tarefa 6, os voluntários tinham alguma dificuldade em encontrar o local onde seria feito um pagamento de forma periódica. O problema talvez tenha sido a palavra "Agendar" presente na descrição da tarefa, pois a maioria dos voluntários começou por tentar entrar nos "Pagamentos agendados" quando o esperado seria entrarem no "Pagamento de serviços" e no botão "Periodicidade" selecionar a opção "Mensal".

- Observação: Quando se elimina o pagamento agendado, deveria aparecer um *popup* para confirmar.

Tarefas afetas: 7

Explicação: Quando um utilizador carrega no botão para eliminar os pagamentos agendados selecionados, deveria aparecer um *AlertDialog*, de modo a que o utilizador confirme a ação.

- Observação: Não está intuitivo que se tem de ficar a carregar no item para eliminar o pagamento agendado.

  Tarefas afetas: 7

  Explicação: A maioria dos voluntários carregaram nos 3 pontos presentes na *toolbar*, esperando encontrar uma opção para eliminar ou selecionar os pagamentos.

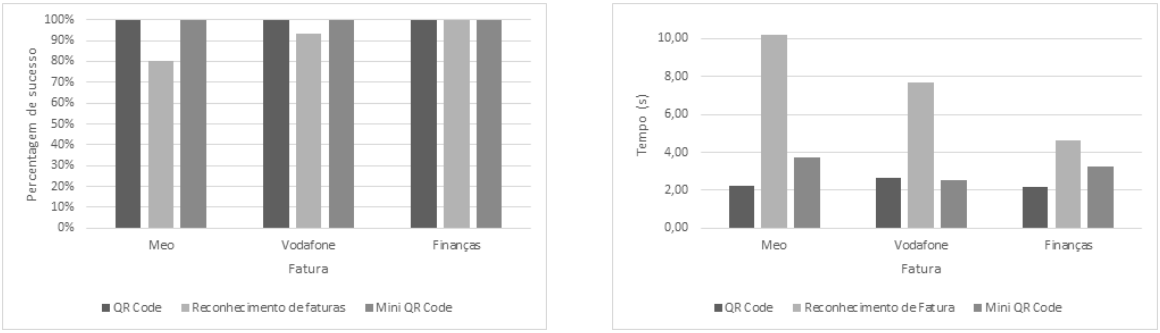
Como já foi mencionado, a tarefa 5 foi a que obteve pior *feedback* por parte dos voluntários. Isto pode ter ocorrido devido à dificuldade deste pagamento e aos conceitos que o mesmo envolve, como por exemplo, o tipo de remuneração e o tipo de pagamento. Além disto, os voluntários também ficaram confusos com a necessidade de se selecionar primeiro o ano e só depois se selecionar o mês, como foi explicado anteriormente.

6.2 Testes do Reconhecimento de Faturas

Existem duas formas de reconhecer uma fatura, a partir do reconhecimento do campo de "Pagamento por Multibanco", a partir de um detetor, ou a partir do reconhecimento de um *QR Code*. O primeiro é algo que pode ser aplicado na prática, já o segundo, é fictício e para ser aplicado iria ser necessário que as faturas emitidas possuíssem um *QR Code*, criado com as normas explicadas na Secção *"QR Code"*, do Capítulo 5.

Foram realizados testes para comparar a eficiência destas duas formas de reconhecimento de uma fatura. Em primeiro lugar, foram testados *QR Codes* nas faturas da Meo, Vodafone e Finanças com dimensões iguais a 3,23 x 3,23 cm, 2,83 x 2,83 cm e 5,29 x 5,29 cm respetivamente. Mais tarde, estes foram diminuídos para o tamanho 1,21 x 1,21 cm, constituindo um segundo reconhecimento por *QR Code*, sendo nomeados como Mini *QR Code*. Por último, o reconhecimento de faturas através do campo "Pagamento por Multibanco" foi, também, testado nas mesmas faturas que os métodos já mencionados. As três faturas de teste são apresentadas no Anexo E. Para cada teste foi anotado o tempo despendido desde o momento em que o utilizador carrega no botão para efetuar o reconhecimento, até ao momento em que a página de "Pagamento de serviços" ou "Pagamento ao estado" é exibida com os campos já preenchidos. Também é testado o sucesso da deteção, ou seja, se os campos do pagamento foram reconhecidos e preenchidos corretamente de acordo com a fatura testada. Para cada fatura e modo de reconhecimento foram feitos 15 testes.

Os resultados obtidos após a realização dos testes foram representados através de dois gráficos. Na Figura 6.3a representa-se a percentagem de sucesso de cada um dos métodos de reconhecimento de fatura. Na Figura 6.3b é representado o tempo despendido na deteção das faturas, novamente através de cada um dos métodos.



(a) Grau de sucesso da deteção

(b) Tempo para deteção

Figura 6.3: Resultados dos testes de reconhecimento de faturas

Na Figura 6.3a é possível verificar que o reconhecimento por *QR Code* não teve falhas no sucesso do reconhecimento, sendo que as falhas neste tipo de reconhecimento apenas ocorrem quando o *QR Code* é gerado com incorreções. Já no reconhecimento do campo de pagamento de multibanco das faturas há um maior risco de haver uma falha, pois as falhas geralmente acontecem quando um número é mal detetado sendo desta maneira confundido com outro. Estas falhas podem ocorrer com frequência em faturas com fraca qualidade de impressão. Além disto, também há o risco de serem reconhecidos outros números na fatura que não os pretendidos. Neste tipo de reconhecimento atingiu-se um sucesso de 100% na fatura das finanças pois o conteúdo a detetar possui dimensões muito superiores às outras duas faturas, podendo não corresponder à realidade. Desta forma, conclui-se que o reconhecimento por *QR Code* possui uma taxa de sucesso superior ao reconhecido através do campo de "Pagamento por Multibanco".

Na Figura 6.3b verifica-se que o tempo despendido para detetar uma fatura, tanto através de um *QR Code*, como de um Mini *QR Code* é substancialmente menor do que o tempo despendido para detetar os dados de uma fatura através do reconhecido através do campo de "Pagamento por Multibanco".

Assim, é possível concluir que seria mais eficiente utilizar o reconhecimento de uma fatura por QR Code. Este seria algo que poderia ser facilmente implementado, tanto por parte dos bancos, como por parte dos softwares de faturação utilizados pelas várias entidades portuguesas.

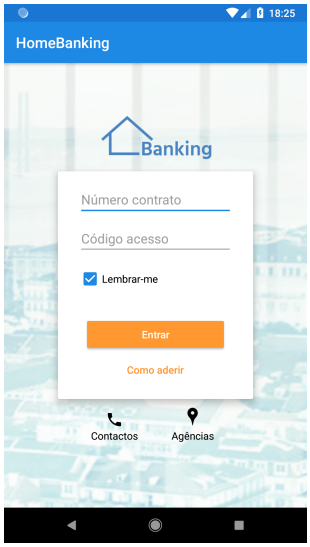
6.3 Testes de Compatibilidade

O sistema operativo Android já se encontra atualmente na oitava versão, sendo que cada versão possui características únicas que por vezes podem levar a falhas de compatibilidade nas aplicações desenvolvidas. Para evitar a ocorrência destas falhas, o nível mínimo de API deve ser declarado, impedindo, assim, que dispositivos com níveis inferiores ao declarado, consigam instalar a aplicação. Além disto, existe um nível alvo de API que também deve ser declarado. Este faz com que o sistema ative comportamentos de compatibilidade quando o nível de API da plataforma for superior ao declarado como alvo na aplicação, garantindo que esta continue a funcionar da forma esperada.

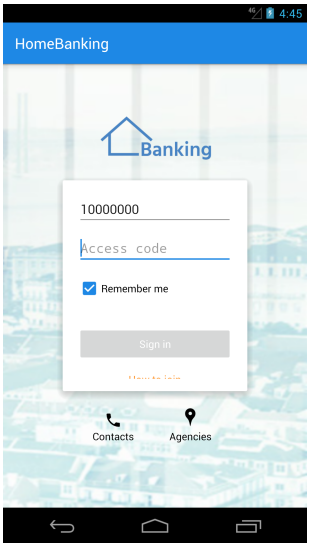
Com isto, foram declarados na aplicação os valores de API 16 e 26 para o valor mínimo e alvo, respetivamente. Para testar a compatibilidade da aplicação esta foi instalada em dois dispositivos, um deles com a versão Android Jelly Bean 4.1 correspondente ao nível mínimo de API da aplicação e outro com a versão Android Oreo 8.1 que é atualmente a mais recente. No dispositivo com a versão mais recente a aplicação funcionou como esperado e não teve qualquer tipo de falha de compatibilidade.

No dispositivo com a versão Android Jelly Bean 4.1 a aplicação funcionou corretamente tendo apenas algumas falhas de compatibilidade relacionadas com os *layouts*, sendo estas as seguintes:

- Comparando a Figura 6.4b com a Figura 6.4a é possível verificar que há uma falha de compatibilidade, pois, na versão inferior, o texto "Como aderir" está cortado e o botão "Entrar" está com uma cor diferente da esperada.
- Na Figura 6.5b é possível verificar que o texto dentro dos botões não está centrado como seria esperado (Figura 6.5a).
- Comparando a Figura 6.6b com a Figura 6.6a, é possível verificar que há uma falha de compatibilidade, pois, na versão inferior, os botões não se encontram na cor esperada e a margem do botão para selecionar a periodicidade também não se encontram de acordo com o esperado.

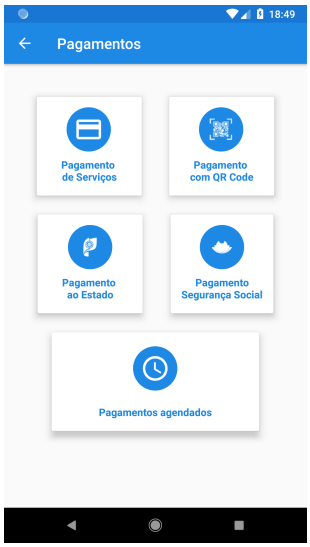


(a) Android Oreo 8.1 (API 27)

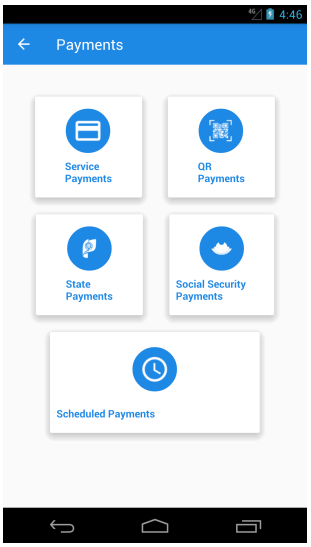


(b) Android Jelly Bean 4.1 (API 16)

Figura 6.4: Ecrã de Login

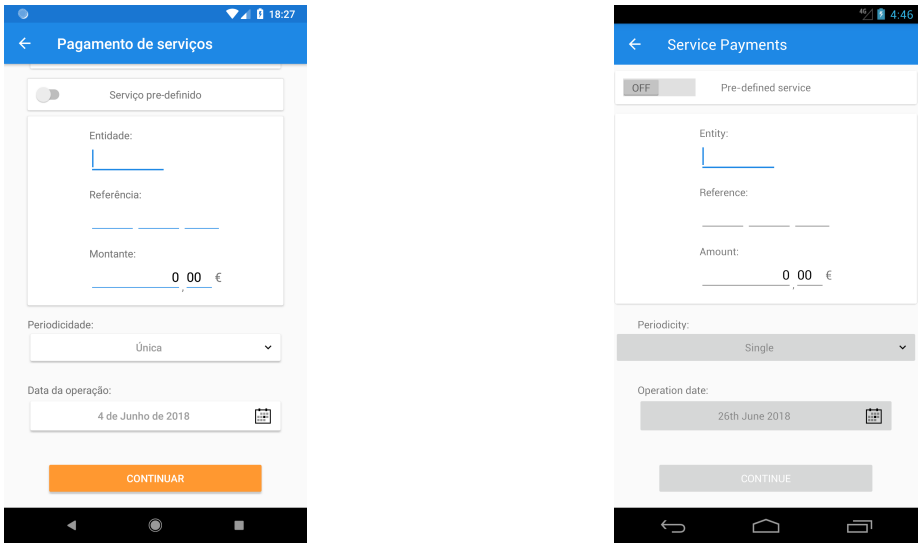


(a) Android Oreo 8.1 (API 27)



(b) Android Jelly Bean 4.1 (API 16)

Figura 6.5: Ecrã do Menu de Pagamentos



(a) Android Oreo 8.1 (API 27)

(b) Android Jelly Bean 4.1 (API 16)

Figura 6.6: Ecrã dos Pagamento de Serviços

Capítulo 7

Conclusão

Este capítulo irá ser dividido em três secções. Na primeira secção começa-se por apresentar o trabalho realizado e as suas finalidades. Na segunda secção são enumeradas as dificuldades encontradas. E na terceira secção são feitas algumas sugestões de trabalho futuro.

7.1 Conclusões Principais

Cada vez existe mais concorrência no que diz respeito a agências bancárias. Para que estas tenham sucesso, têm de oferecer os melhores serviços, não só a nível financeiro mas também a nível informático, de forma a oferecer as melhores comodidades aos seus clientes.

Hoje em dia já todos os bancos possuem uma aplicação móvel e a qualidade da mesma é um fator diferenciador. Esta pode também constituir um fator decisivo para os clientes, quando estes se encontram num processo de seleção de um banco.

Neste projeto foi desenvolvida uma aplicação com o objetivo atingir uma qualidade superior à existente no mercado. Um dos objetivos desta aplicação foi esta servir de ponto de partida para aplicações reais de *homebanking*, sendo apenas necessário completá-la com as funcionalidades necessárias seguindo o mesmo padrão arquitetural.

A aplicação desenvolvida seguiu um padrão arquitetural MVP, facilitando, desta forma, a perceção e a modificação do código por parte dos programadores. Esta foi desenvolvida na linguagem Java e, mais tarde, foi realizada a migração da mesma para a linguagem Kotlin. O suporte desta linguagem, em fóruns da comunidade, ainda não é tão vasto como da linguagem Java. Contudo, decidiu-se fazer à mesma a migração, pois é uma linguagem que se encontra em grande ascensão. Num questionário anual feito pela *stackoverflow*, Kotlin foi considerada a linguagem mais adorada em 2018 [12], confirmando assim a grande probabilidade de esta vir a ser uma das linguagens mais utilizadas no futuro.

Na aplicação foi implementada uma ideia inovadora: reconhecimento de faturas a partir de um *QR Code*. Esta ideia foi testada e comparada com o reconhecimento de faturas a partir do campo "Pagamento por Multibanco" (tecnologia já existente no mercado) e os melhores resultados foram obtidos pela nova tecnologia. Desta forma, introduzir este novo reconhecimento de faturas no mercado iria trazer vantagens, sendo que a implementação desta seria bastante fácil e de baixo custo.

Também foram feitos testes de usabilidade que confirmaram a satisfação dos utilizadores quanto à interface e experiência de utilização. Uma observação feita por grande parte dos utilizadores foi que esta

aplicação é mais intuitiva e visualmente mais agradável do que aplicação que os próprios utilizam para aceder à conta do seu banco.

Assim, pensa-se que os objetivos iniciais definidos para o projeto foram superados.

7.2 Dificuldades Encontradas

Ao longo do estágio foram encontradas várias dificuldades, tais como:

Alteração do plano inicial: Era esperado, inicialmente, integrar uma equipa de trabalho mas infelizmente isso não aconteceu. Por esse motivo não foi possível adquirir conhecimento mais especializado com programadores seniores. Sendo que, durante o estágio, os conhecimentos técnicos foram adquiridos a partir de uma vasta pesquisa.

Utilização do padrão MVP: A utilização deste padrão de desenho mantém o programa limpo e fácil de compreender. Apesar disso a implementação do mesmo trás algumas dificuldades, pois a separação entre a camada *View* e a camada *Presenter* nem sempre é trivial. Por vezes foi necessário criar *listeners* para obter a informação das ações do utilizador, sendo que a lógica de implementação destes não é fácil numa fase inicial. Por vezes também foi necessário obter valores dos ficheiros strings no lado da camada *Presenter*, como estes só são possíveis de utilizar recorrendo ao *Context* e as boas práticas dizem que só se deve utilizar o *Context* do lado da *View*, foi necessário alterar a lógica do código de maneira a que não fosse necessário utilizar o *Context* na camada *Presenter*.

Notificações para grupos: Quando as notificações foram implementadas, houve a tentativa de implementar notificações para grupos de dispositivos, essa tentativa não teve sucesso devido à complexidade da sua implementação. Após sucessivos erros, decidiu-se não perder mais tempo com esta funcionalidade.

Migração para Kotlin: O Android Studio tem ferramentas muito boas para a conversão do código Java para Kotlin. Este ponto constituiu uma dificuldade pois cometeu-se um erro de principiante ao não fazer a conversão do código por ordem crescente de dependências. Isto levou a que as classes com código Kotlin gerassem alguns erros quando tinham vários dependentes, isto porque, apesar de a linguagem Kotlin ser interoperável com a linguagem Java, as conversões por vezes precisavam de algumas alterações para que estas funcionassem corretamente entre si. Assim os testes das conversões foram dificultados.

7.3 Trabalho Futuro

Após a conclusão deste projeto, ainda existem funcionalidades que podem ser melhoradas. Desta forma, sugere-se o seguinte como trabalho futuro:

- Realização das restantes funcionalidades, como saldos e movimentos, posição integrada, consultas, entre outras, de maneira a obter uma aplicação de *homebanking* completa.
- Filtrar os pagamentos agendados por conta, como, por exemplo, se faz na funcionalidade “Pagamentos Frequentes”. Na funcionalidade “Pagamentos Agendados”, são apresentados todos os

pagamentos de todas as contas, podendo assim dificultar a procura de um pagamento agendado caso o utilizador possua um número considerável de agendamentos.

- O saldo disponível e o saldo contabilístico é sempre apresentado ao utilizador, seja num multibanco ou numa aplicação de *homebanking*, ao contrário desta aplicação que apenas apresenta o saldo disponível.
- As notificações para grupos de dispositivos possibilita ao utilizador receber notificações de uma aplicação em todos os dispositivos com essa aplicação instalada. Isto não foi realizado nesta aplicação e, portanto, o utilizador só receberá notificações no ultimo dispositivo onde a aplicação foi instalada.
- Poderia existir um modo *offline*, permitindo assim ao utilizador efetuar um pagamento sem a necessidade de estar conectado à Internet. Seriam guardados os dados do pagamento e quando o utilizador voltasse a estar online, o pagamento seria efetuado. Esta é uma funcionalidade que pode ser impossível na prática por motivos de segurança.
- Há a necessidade de haver um melhoramento da segurança da aplicação, principalmente na matriz de autenticação que deveria ser encriptada.
- Por fim, os erros encontrados durante a fase de testes de utilização devem ser resolvidos.

Bibliografia

[1] android.hardware.fingerprint — android developers. https://developer.android.com/reference/android/hardware/fingerprint/package-summary. (Último acesso: 30 de Novembro 2018).

[2] Aplicação móvel da caixa geral de depósitos - caixadirecta. https://play.google.com/store/apps/details?id=pt.cgd.caixadirecta. (Último acesso: 7 de Junho de 2018).

[3] Aplicação móvel do montepio - app m24. https://www.montepio.pt/app-m24-particulares. (Último acesso: 7 de Junho de 2018).

[4] Aplicação móvel do novo banco - nb smart app. https://www.novobanco.pt/site/cms.aspx?labelid=nbsmartapp. (Último acesso: 7 de Junho de 2018).

[5] Github - conjunto de amostras de diferentes arquiteturas e padrões para aplicações android. https://github.com/googlesamples/android-architecture/. (Último acesso: 4 de Junho de 2018).

[6] Github - impressão digital para android. https://github.com/googlesamples/android-FingerprintDialog. (Último acesso: 12 de Junho de 2018).

[7] Google anuncia kotlin como linguagem oficial para desenvolvimento android na conferência google i/o 2017. https://www.youtube.com/watch?v=d8ALcQiuPWs&t=2s. (Último acesso: 21 de Junho de 2018).

[8] Java persistence api. https://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. (Ultimo acesso: 30 de Novembro de 2018).

[9] Javamailsender (5.0.7.release api). https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/mail/javamail/JavaMailSender.html. (Último acesso: 21 de Julho de 2018).

[10] Jparepository (spring data jpa 2.0.8.release api). https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html. (Último acesso: 21 de Julho de 2018).

[11] Mvc vs. mvp vs. mvvm no android. https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/. (Último acesso: 4 de Junho de 2018).

[12] Stack overflow questionário feito a programadores - 2018. https://insights.stackoverflow.com/survey/2018/. (Último acesso: 23 de Julho de 2018).

[13] String (biblioteca java). https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-. (Último acesso: 19 de Junho de 2018).

[14] Android. Introdução ao android studio. https://developer.android.com/studio/intro/. (Último acesso: 11 de Maio de 2018).

[15] Millennium BCP. Definição de iban. http://millenniumbcp.custhelp.com/app/answers/detail/a_id/27/~/o-que-%C3%A9-o-nib-e-o-iban%3F. (Último acesso: 8 de Junho de 2018).

[16] Android Developers. Painéis. https://developer.android.com/about/dashboards/. (Último acesso: 25 de Junho de 2018).

[17] Google Developers. Api para deteção de códigos de barras e qr codes - mobile vision. https://developers.google.com/vision/android/barcodes-overview. (Último acesso: 19 de Junho de 2018).

[18] Google Developers. Api para reconhecimento de texto - mobile vision. https://developers.google.com/vision/android/text-overview. (Último acesso: 20 de Junho de 2018).

[19] Microsoft Docs. Firebase cloud messaging. https://docs.microsoft.com/pt-br/xamarin/android/data-cloud/google-messaging/firebase-cloud-messaging. (Último acesso: 29 de Junho de 2018).

[20] Firebase. Firebase cloud messaging. https://firebase.google.com/docs/cloud-messaging/. (Último acesso: 21 de Junho de 2018).

[21] David Garlan, Felix Bachmann, James Ivers, Judith Stafford, Len Bass, Paul Clements, and Paulo Merson. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2nd edition, 2010.

[22] Kotlin. Null safety. https://kotlinlang.org/docs/reference/null-safety.html. (Último acesso: 22 de Junho de 2018).

[23] Basil Miller. Github - pulseview: Ferramenta que gera pulsação em torno de icons. https://github.com/Devlight/PulseView. (Último acesso: 14 de Junho de 2018).

[24] moondroid. Github - coverflow: Ferramenta para fluxo de capas para android. https://github.com/moondroid/CoverFlow. (Último acesso: 18 de Junho de 2018).

[25] Multibanco. Descrição dos vários tipos de pagamento. https://www.multibanco.pt/operacoes/pagamentos/. (Último acesso: 11 de Maio de 2018).

[26] ProAndroidDev. Transformação de mvp para mvvm. https://proandroiddev.com/mvp-to-mvvm-transformation-611959d5e0ca. (Último acesso: 4 de Junho de 2018).

[27] Equation Research. What users want from mobile. Research, Compuware, July 2011.

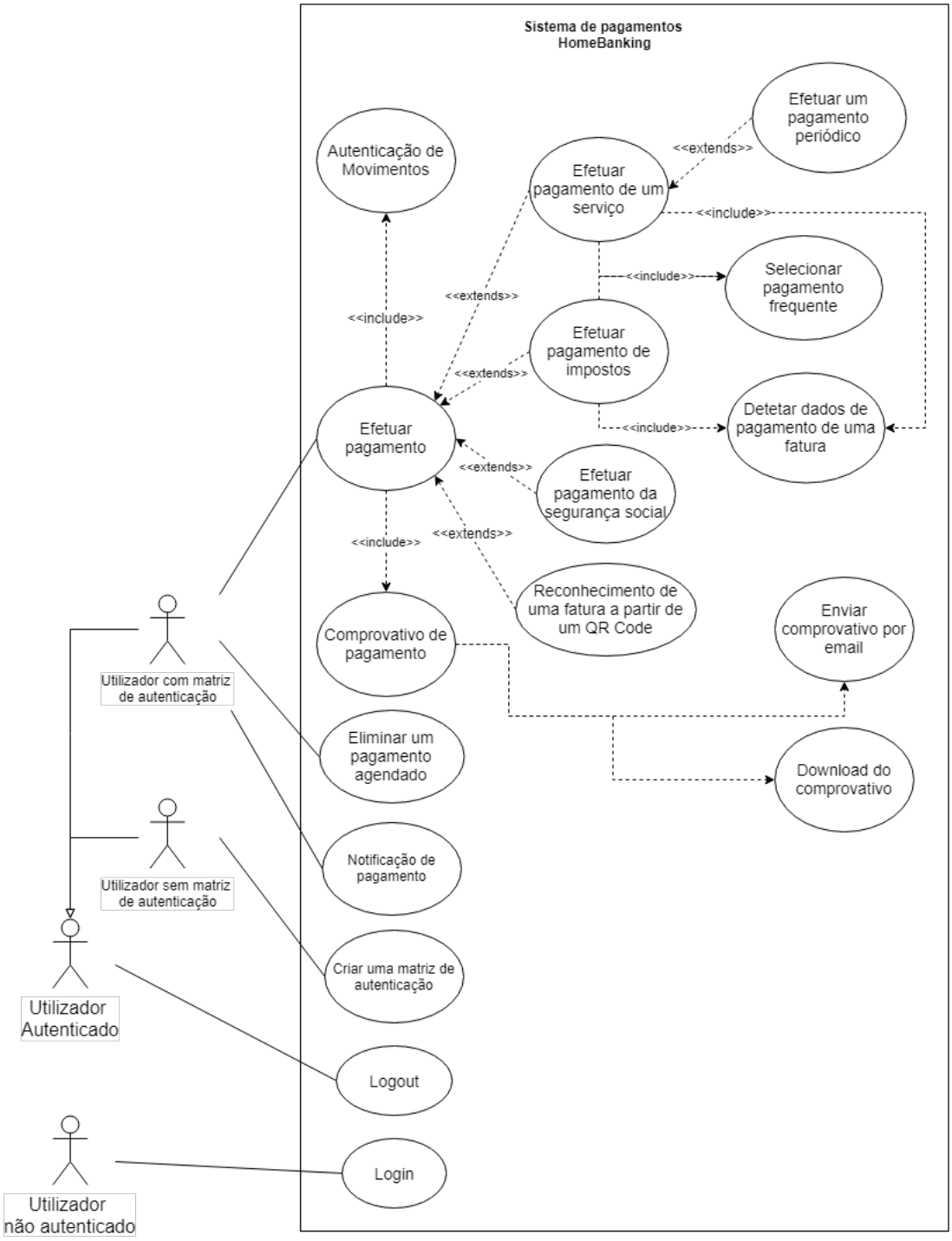
[28] Scientiamobile. Mobile overview report. Research, Scientiamobile, March 2017.

[29] Segurança Social. Cálculo das contribuições. http://www.seg-social.pt/contribuicoes. (Último acesso: 18 de Junho de 2018).

[30] Margaret Rouse WhatIs.com. Two-factor authentication (2fa). https://searchsecurity.techtarget.com/definition/two-factor-authentication. (Último acesso: 1 de Junho de 2018).

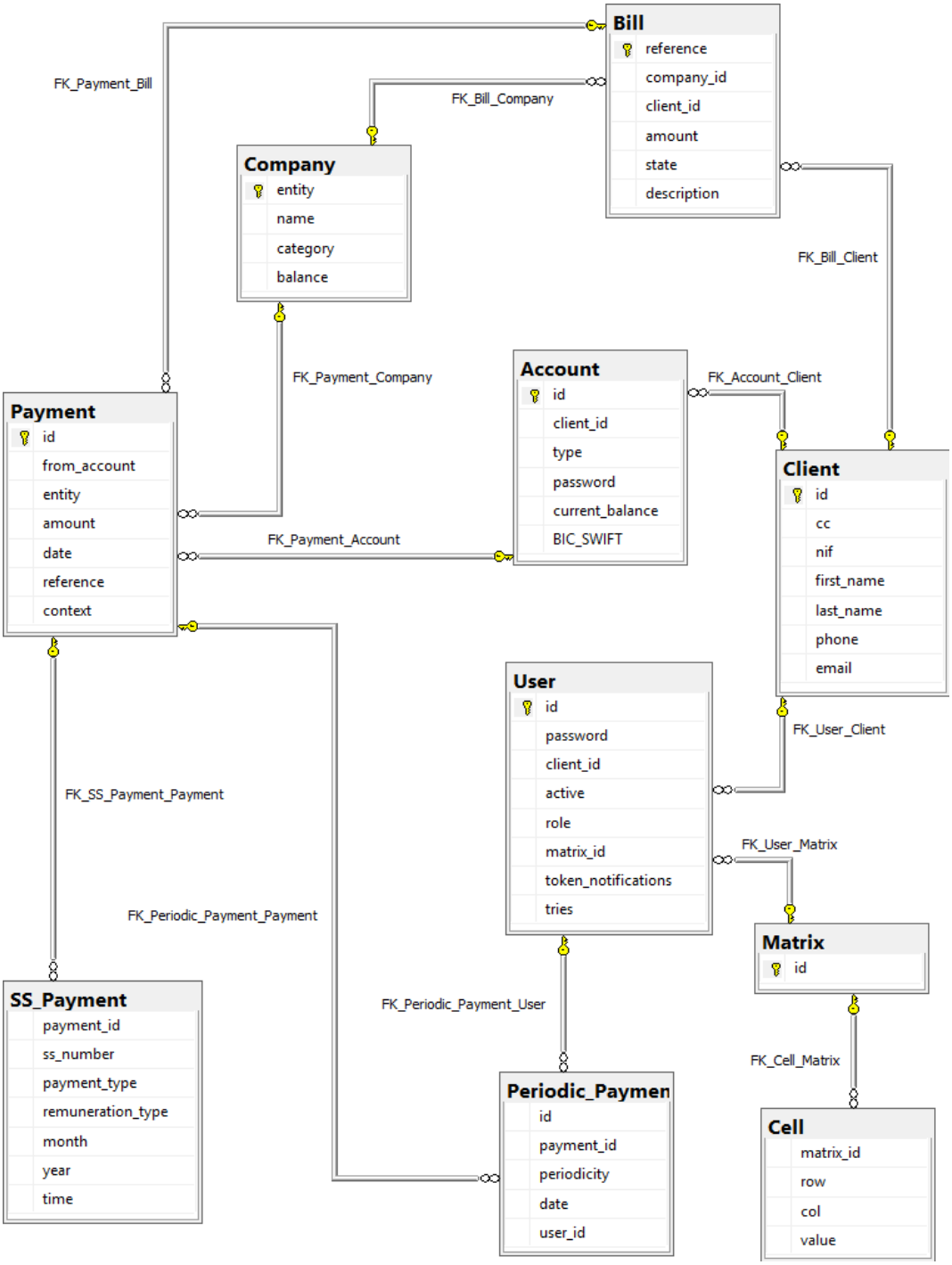
Anexo A

Diagrama de Casos de Uso



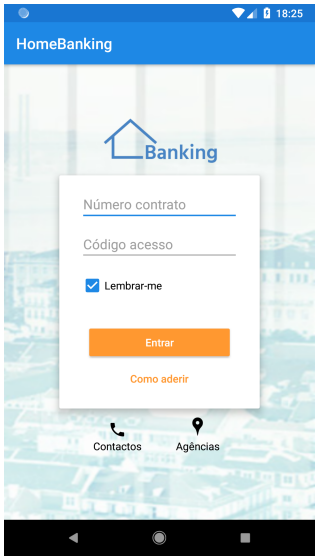
Anexo B

Diagrama de Base de Dados

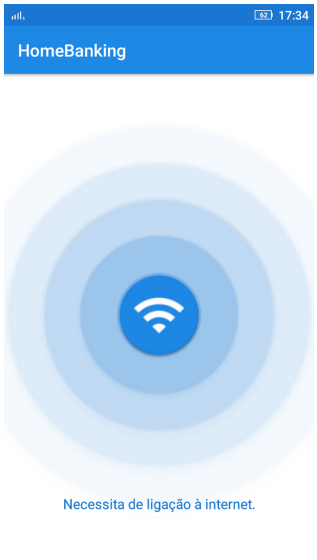


Anexo C

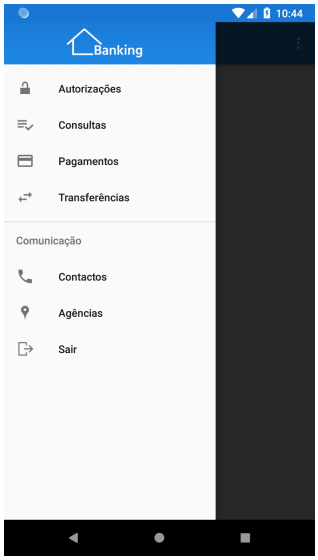
Layouts da aplicação



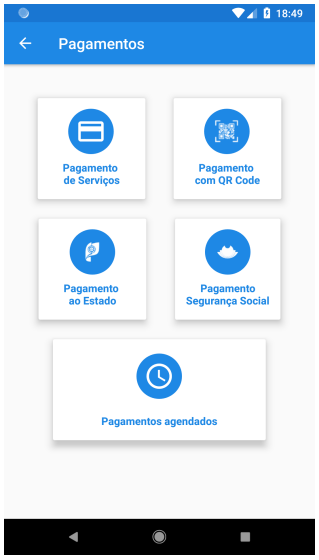
(a) Login



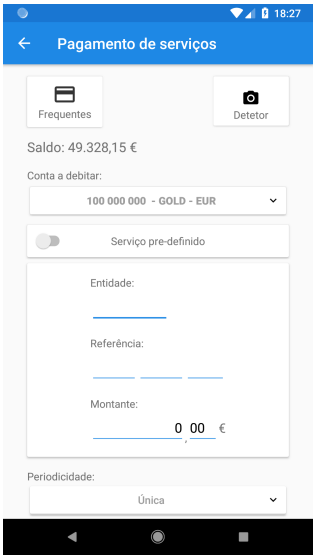
(b) Vista offline



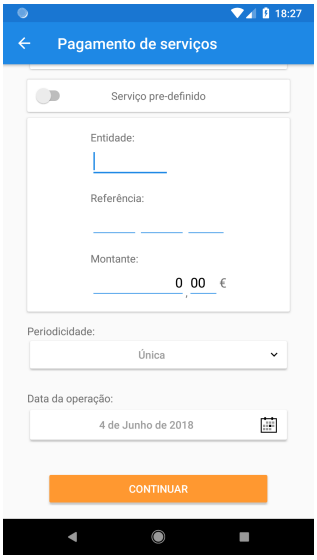
(c) Menu hambúrguer



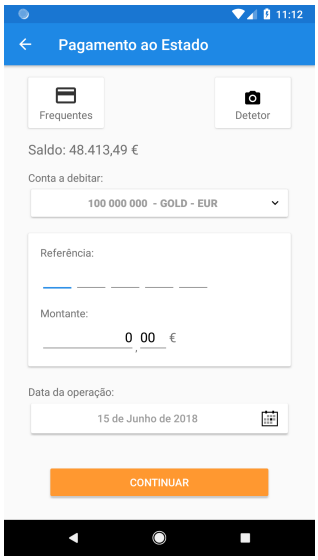
(d) Menu de pagamentos



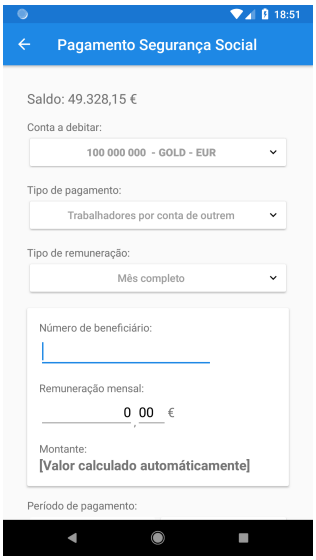
(e) Pagamento de serviços



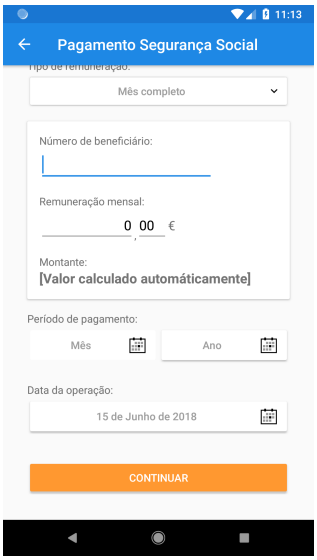
(f) continuação



(a) Pagamento ao estado

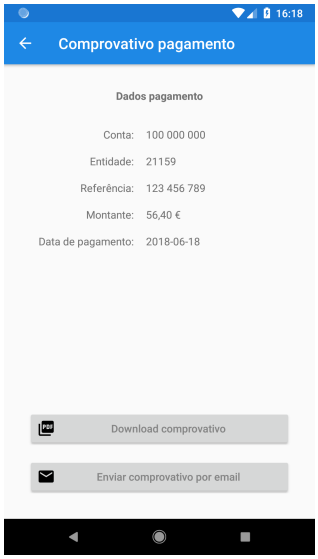


(b) Pagamento segurança social

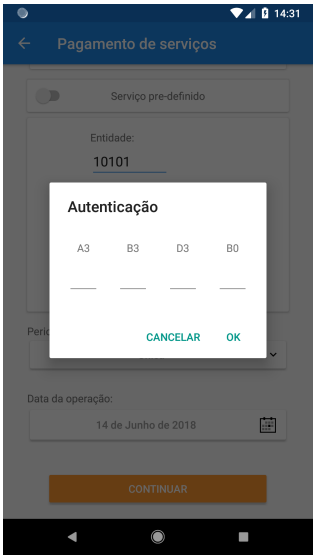


(c) continuação

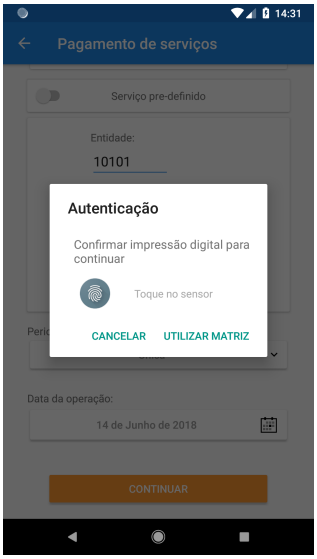
Figura C.2: Layouts finais da aplicação



(a) Comprovativo de pagamento



(b) Autenticação com matriz



(c) Autenticação Impressão Digital

Figura C.3: Layouts finais da aplicação

Anexo D

Emails e Comprovativo de Pagamento

D.1 Email após a Criação de uma Matriz de Autenticação

Rodrigo Rodrigues <rodrigomtr95@gmail.com>

HomeBanking
1 mensagem

---

banking.payments.app@gmail.com <banking.payments.app@gmail.com> 14 de junho de 2018 às 12:20
Para: rodrigomtr95@gmail.com

Olá, Rodrigo Rodrigues

Agora pode fazer pagamentos na aplicação HomeBanking!
A sua matriz de autenticação é a seguinte:

```
   0 1 2 3 4
  =========
A | 1 2 6 0 4
B | 5 4 7 1 1
C | 9 2 6 4 9
D | 8 5 4 0 4
E | 2 9 5 4 6
```

Aceda ao seu banco onde, como e quando quiser!

D.2 Email de Comprovativo de Pagamento

Rodrigo Rodrigues <rodrigomtr95@gmail.com>

Envio de comprovativo HomeBanking
1 mensagem

banking.payments.app@gmail.com <banking.payments.app@gmail.com> 21 de julho de 2018 às 11:08
Para: rodrigomtr95@gmail.com

Exmo(a) Senhor(a) Rodrigo

Enviamos em anexo o comprovativo de registo da operação de Pagamento, efetuada no serviço Homebanking App.

Para qualquer esclarecimento sobre esta comunicação poderá contactar o ordenante da operação.

2035.pdf
15K

D.3 Email de Falha do Pagamento Agendado

Rodrigo Rodrigues <rodrigomtr95@gmail.com>

Envio de comprovativo HomeBanking
1 mensagem

banking.payments.app@gmail.com <banking.payments.app@gmail.com> 1 de agosto de 2018 às 21:14
Para: rodrigomtr95@gmail.com

Exmo(a) Senhor(a) Rodrigo

Na sequência da operação de Pagamento agendado efetuada no serviço Homebanking App, Informamos que não foi possível efetuar o seguinte pagamento:
Entidade: 10297
Referência: 249332561
Montante: 71.85

Para qualquer esclarecimento sobre esta comunicação poderá entrar em contacto com a agência.

SLOGAN

D.4 Comprovativo de Pagamento

Comprovativo de pagamento

Exmo(a) Senhor(a)
Na sequência do pedido efetuado por Rodrigo Rodrigues, o serviço HomeBanking registou a operação abaixo referida.

Detalhes

Conta	100 000 010
Entidade	10297
Referência	249 332 561
Montante	71,85 €
Data de pagamento	2018-06-18

Para mais informações poderá contactar o serviço Homebanking através dos telefones 707 XX XX XX - 21 XXX XX90 - 91 XXX XX XX - 93 XXX XX XX - 96 XXX XX XX (24 horas por dia/todos os dias do ano), no nosso site www.homebanking.pt/ em "Espaço Cliente" ou em qualquer Agência.

SLOGAN

Anexo E

Faturas Fictícias

Nas páginas seguintes serão apresentadas 3 faturas fictícias:

- Fatura MEO;
- Fatura Vodafone;
- Fatura Finanças.

Estas foram utilizadas durante a fase de testes à aplicação.

MEO

Área de Cliente em meo.pt

Apoio ao cliente e faturação
16 96 (MEO rede móvel)
961 000 033 (outras redes)

altice

Fatura

Nº Fatura:	**FT MV/123456789**
Nº Referência:	999999999
Data de Emissão:	2016-06-03
Nº Contribuinte:	123123123
Nº Conta:	9876543210
Nº Telefone:	961234567
Tarifário	Unlimited L
Pacote Adquirido	

2016-01-29 1-C6

ZEFERINO MEIRELLES FAGUNDES

R PAPOILAS 66
RC
FATAÚNÇOS
6666 - 123 FATAÚNÇOS

Utilizador ZEFERINO MEIRELLES FAGUNDES

Fatura junho 2016

Data Limite de Pagamento 2016 - 06-27

Valor a Pagar € 56,40

Fatura	Qtd (uni)	Valor (€ s/IVA)	IVA (%)	IVA (€)	Valor (€ c/IVA)
SERVIÇOS DE COMUNICAÇÕES					
Mensalidades	1	20,454	23,00	4,70	25,154
Consumos	1	9,081	23,00	2,09	11,171
Arredondamento		0,005			
Total da fatura		29,540		6,79	**36,33**

Extrato de Conta	Valor (€ c/IVA)
Saldo anterior	20,070
Fatura nº FT MV/123456789	36,330
Valor a Pagar	**56,40**

WAITING RING: Dê música a quem lhe liga, enquanto esperam que atenda!
Se gosta de Ellie Goulding, ligue 12702 e aproveite oferta de música ON MY MIND até 31-01. Há milhares de êxitos disponiveis para escolher os seus favoritos.
O primeiro mês é gratuito para novas adesões. Saiba mais em waiting ring. meo.pt

Adira ao Débito Direto na sua Área de Cliente

A adesão ao débito direto é gratuita e permite pagar as suas faturas por transferência bancária na sua conta.
Deste modo evita filas e preocupações com datas limites de pagamento.
O valor a pagar bem como a data de pagamento é comunicado com antecedência na sua fatura.

Pagamento por Cheque

Nº de Conta 123456789

O cheque deve ser emitido à ordem da MEO - Serviços de Comunicações e Multimédia S.A., e enviado com este destacável para: MEO - Serviços de Comunicações e Multimédia S.A., Apartado 21105, EC Terreiro do Paço, 1128-001 Lisboa.
Por favor escreva no verso do cheque o seu nº de conta.

Pagamento por Multibanco

MB MULTIBANCO

Entidade	211 59
Referência	123 456 789
Montante	€ 56,40

O talão emitido pelo caixa automático faz prova de pagamento. Conserve-o.

Utilize sempre esta Entidade e Referência para pagamento.

Pagamento por QR Code:

Pagamento Payshop

00000000000000000000000000000000000000000000000000

00000000000000000000 000 000000000 00

+ RESERVADO A MARCAÇÃO ÓTICA, NÃO ESCREVER, NÃO CARIMBAR, NÃO AGRAFAR +

00000000000000000000 000 000000000 00

vodafone

Fatura **dez 2015**

Nº Documento	Nº Contribuinte	Nº de Conta
123123123	123123123	300xxxxx

Apoio a Clientes - Visite ajuda.vodafone.pt ou ligue 16912 (24 horas por dia)
Apoio técnico - Ligue 16913 (das 7h às 2h)
My Vodafone - Faça a gestão dos serviços, consulte e pague faturas em my.vodafone.pt

Tem 3635 pontos Clube Viva
50 pontos vão perder a validade a 31 de janeiro de 2016

Mensagens importantes

1 O valor mensal do seu Tv Net Voz já inclui um desconto que será aplicado durante 2 anos.

[redacted]

Pagamento por QR Code:

Valor deste mês

Período de faturação: **1 dez a 31 dez** Data de emissão: **5 jan 2016**

MB MULTIBANCO
Entidade: 10297
Referência: 249 332 561
Montante: € 71,85

€71,85

Data limite de pagamento: **23 jan 2016**

Valores com IVA incluído	IVA	Mensalidade	Serviços Suplementares	Comunicações	Encargos Adicionais	Deduções	Total
Tv Net Voz 211913XXXX	23%	€25,90	–	–	–	–	**€25,90**
Red 919200XXXX	23%	€25,90	–	–	–	–	**€25,90**
Red 91923XXXX	23%	€9,90	–	–	–	–	**€9,90**
Red 91925XXXX	23%	€9,90	–	€0,25	–	–	**€10,15**
						Total com IVA	**€71,85**

Processado por computador

Respeite a data limite de pagamento e evite a suspensão do serviço

Após a data limite de pagamento, o seu serviço pode ser suspenso e só poderá voltar a usá-lo depois de pagar o valor em atraso. Se o valor em atraso continuar por pagar depois da suspensão, o seu contrato poderá ser anulado. Consulte as condições em vodafone.pt ou numa das nossas lojas.

Resumo do IVA

Taxa IVA	Valor Base	Valor IVA
23%	€58,41	€13,44

Este é um documento fictício

AT autoridade tributária e aduaneira	**[DESCRIÇÃO DO PAGAMENTO]**
IDENTIFICAÇÃO DO DOCUMENTO	**INDENTIFICAÇÃO FISCAL**
20187093 De: 2018	[nif]
DATA LIMITE DE PAGAMENTO	**MORADA**
[data limite do pagamento]	[morada]

Referência para Pagamento
123 123 123 123 123
Importância a pagar
100,00 €

O pagamento pode ser efetuado através do Multibanco, da Internet, dos CTT, das Instituições de Crédito e dos Serviços de Finanças (Secções de Cobrança), utilizando a referência indicada.

Para efetuar o pagamento através da Internet utilize o serviço on-line do seu Banco e selecione Pagamento ao Estado, ou pagamento por QR Code, utilizando o código abaixo.

QR Code:

Glossário

2FA Autenticação com dois fatores (Two Factor Authentication).

AlertDialog Pequenas janelas que levam o utilizador a tomar uma decisão ou inserir informações adicionais. Não ocupam a tela toda e são normalmente utilizadas para que os utilizadores realizem uma ação antes de continuar..

CardView Componente de design que faz um efeito de carta, servindo para organizar outras componentes de design dentro desta..

CGD Caixa Geral de Depósitos.

EditText Campo para o preenchimento de texto numa aplicação Android..

FCM Tecnologia da Firebase para o envio de notificações entre diferentes plataformas (Firebase Cloud Messaging).

Home banking realização de operações bancárias sem a necessidade de deslocação a um banco ou multibanco..

HTTP Protocolo de comunicação na rede (HyperText Transfer Protocol).

IDE Aplicação com várias funcionalidades que ajudam os programadores a desenvolverem software (Integrated Development Environment)..

JDBC É uma interface de programação na linguagem Java que define um padrão de acesso à base de dados (Java DataBase Connectivity).

JSON Formato para a representação de dados a serem tranferidos (JavaScript Object Notation).

JVM Máquina virtual Java (Java Virtual Machine).

Listener Classe que contém uma interface com um único método que é chamado quando um certo evento acontece. Esta classe permite detetar ações do utilizador..

Mocks Em programação orientada a objetos, mocks são objetos que simulam o comportamento de objetos reais, são normalmente utilizados para testar o comportamento de outro objeto..

MVC Model-View-Controller.

MVP Model-View-Presenter.

MVVM Model-View-ViewModel.

NISS Número de Inscrição na Segurança Social.

Programação assíncrona Programação realizada de maneira a que o programa não necessite de esperar que uma tarefa termine para executar a próxima.

Ps Programação realizada de maneira a que o programa execute uma tarefa de cada vez, esperando que a tarefa em execução termine para poder começar a seguinte.

Push notifications Notificações que aparecem no smartphone. Ex.: Notificação da receção de uma nova SMS..

Regex Expressão Regural, serve para identificar cadeias de caracteres.

REST Arquitetura que define um conjunto de restrições para a transferência dados sob o protocolo HTTP (REpresentional State Transfer).

SharedPreferences Serve para guardar conjuntos chave-valor numa aplicação Android..

SMTP Protocolo para padronizar a transferência de correio eletrónico (Simple Mail Transfer Protocol).

SOA Arquitetura Orientada a Serviços (Service Oriented Architecture).

TextView Componente de design da aplicação Android responsável pela exibição de texto.

Token conjunto único de caracteres que servem de identificador.

Toolbar Barra de ferramentas da aplicação Android, é nesta que se encontra o botão do menu hambúrguer..